
Parsing Motion and Composing Behavior for Semi-Autonomous Manipulation

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation von Rudolf Lioutikov aus Gornoe, Kasachstan
Tag der Einreichung: 31.07.2018, Tag der Prüfung: 01.10.2018
Darmstadt — D 17

1. Gutachten: Prof. Dr. Jan R. Peters
2. Gutachten: Prof. Ken Goldberg, Ph.D.



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Intelligente Autonome Systeme

Parsing Motion and Composing Behavior for Semi-Autonomous Manipulation

Genehmigte Dissertation von Rudolf Lioutikov aus Gornoe, Kasachstan

1. Gutachten: Prof. Dr. Jan R. Peters
2. Gutachten: Prof. Ken Goldberg, Ph.D.

Tag der Einreichung: 31.07.2018

Tag der Prüfung: 01.10.2018

Darmstadt — D 17

Bitte zitieren Sie dieses Dokument als:

URN: [urn:nbn:de:tuda-tuprints-91148](https://nbn-resolving.org/urn:nbn:de:tuda-tuprints-91148)

URL: <http://tuprints.ulb.tu-darmstadt.de/9114>

Dieses Dokument wird bereitgestellt von tuprints,

E-Publishing-Service der TU Darmstadt

<http://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de



Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

Namensnennung – Keine kommerzielle Nutzung – Weitergabe unter gleichen Bedingungen – 4.0 International

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 31.07.2018



(R. Lioutikov)



Abstract

Robots are becoming an ever bigger part of our day to day life. They take up simple tasks in households, like vacuum cleaning and lawn mowing. They ensure a steady and reliable process at many work places in large scale manufacturing, like the automotive and electronics industry. Furthermore, robots are becoming more and more socially accepted, for instance as autonomous drivers. They even start to engage in special and elderly care, aiming to fill a void created by a rapidly aging population. Additionally, the increasing complexity and capability of robotic systems allows to solve ever more complicated tasks in increasingly difficult scenarios and environments. Soon, encountering and interacting with robots will be considered as natural as interacting with other humans.

However, when it comes to defining and understanding the behavior of robots, experts are still necessary. Robots usually follow predefined routines which are programmed and tuned by people with years of experience. Unintended behavior is traced back to a certain part of the source code which can be modified using a specific programming language. Most of the people that will interact with robotic servants or coworkers in the future, will not have the necessary skill set to instruct robots in such detail. This need for an expert represents a significant bottleneck to the deployment of robots as our everyday companion in households and at work. This thesis presents several novel approaches aiming at facilitating the interaction between non-expert humans and robots in terms of intuitive instruction and simple understanding of the robot capabilities with respect to a given task.

Chapter 3 introduces a novel method that segments unlabeled demonstrations into sequence of movement primitives while simultaneously learning a movement primitive library. This method allows the non-expert to teach an entire task rather than every single primitive. Movement primitives represent a simple, atomic and commonly parameterized motion. The presented method segments each demonstration by identifying similar patterns across all demonstrations and treating them as samples drawn from a learned probabilistic representation of a movement primitive. The method is formulated as an expectation-maximization approach and was evaluated in several tasks, including a chair assembly and segmenting table tennis demonstrations.

In Chapter 4 the previously segmented demonstrations and the learned primitive library are used to induce a formal grammar for movements. Formal grammars are a well established concept in formal language theory and have been applied in several fields, reaching from linguistics, over compiler architecture to robotics. The simplest class of grammars, regular grammars, correspond in their probabilistic form to Hidden Markov Models. However, the intuitive, hierarchical representation of transitions as a set of rules makes it easier for non-experts to comprehend the possible behaviors the grammar implies. A sequence of movements can now be considered a sentence produced by the learned grammar. The production of each sentence can be illustrated

by a tree structure, allowing an easy understanding of the involved rules. Probabilistic context-free grammars are a superset of regular grammars and, hence, are more expressive and exceed the capabilities of Hidden Markov Models. While the induction of probabilistic context-free grammars is considered a difficult, unsolved problem for natural languages, the observed sequences of movement primitives show much simpler structures, making the induction more feasible. The method was successfully evaluated on several tasks, such as a pick-and-place task in a tic-tac-toe setting or a handover task in a collaborative tool box assembly.

Chapter 5 introduces the concept of reinforcement learning into the domain of formal grammars. Given an objective, we apply a natural policy gradient approach in order to learn the grammar parameters that produces sequences of primitives that solve that objective. This allows the autonomous improvement of robot behavior. For instance, a cleaning up task can be optimized for efficiency while avoiding self collisions. The parameters of the grammar are the probabilities of each production. Therefore, probability constraints have to be maintained while learning the parameters. The applied natural policy gradient method ensures reasonably small parameter updates, such that the grammar probabilities change gradually. We derive the natural policy gradient method for formal grammars and evaluate the method on several tasks.

Together, the individual contributions presented in this thesis form an imitation learning pipeline that facilitates the instruction, interaction and collaboration with robots. Starting from unlabeled demonstrations, an underlying movement primitive library is learned while simultaneously segmenting the given demonstrations into sequences of primitives. These sequences are then used to induce a formal grammar. The structure of the grammar and the produced parse trees form a comprehensible representation of the robot capabilities with respect to the demonstrated task. Finally, a reinforcement learning approach allows the autonomous optimization of the grammar given an objective.

Zusammenfassung

Roboter sind ein immer größer werdender Teil unseres täglichen Lebens. Sie übernehmen einfache Aufgaben im Haushalt, wie Staubsaugen und Rasenmähen. Sie ermöglichen einen stabilen und zuverlässigen Ablauf an vielen Arbeitsplätzen in Großbetrieben, wie in der Automobil und Elektronik Branche. Darüber hinaus werden Roboter in der Gesellschaft immer mehr akzeptiert, wie zum Beispiel Systeme für das autonome Fahren. Roboter fangen sogar an in die Betreuung von Senioren und Bedürftigen vorzudringen, um eine Lücke zu schließen die durch eine schnell alternde Gesellschaft geschaffen wird. Zusätzlich erlauben die steigende Komplexität und Leistungsfähigkeit moderner Robotersysteme immer schwerere Aufgaben in immer komplizierter werdenden Szenarien und Umgebungen zu lösen. Bald wird das Antrreffen und Interagieren mit Robotern als so natürlich angesehen wie das interagieren mit Menschen.

Wenn es hingegen darum geht das Verhalten von Robotern zu definieren und zu verstehen, werden immer noch Experten benötigt. Roboter folgen üblicherweise vordefinierten Routinen, die von Menschen mit jahrelanger Erfahrung programmiert und abgestimmt wurden. Unbeabsichtigtes Verhalten wird zu einer bestimmten Stelle im Quellcode zurückverfolgt und kann durch das anwenden einer bestimmten Programmiersprache angepasst werden. Die meisten Menschen, die in der Zukunft mit Roboter-Haushältern und -Mitarbeitern werden interagieren werden, werden jedoch nicht über die benötigten Fähigkeiten verfügen um Roboter in solchem Detail zu instruieren. Diese Abhängigkeit von Experten stellt eine wesentliche Hürde für den Einsatz von Robotern als unsere täglichen Begleiter im Haushalt und auf der Arbeit da. Diese Dissertation präsentiert mehrere neue Ansätze, die das Ziel verfolgen die Interaktion zwischen Nicht-Experten und Robotern. Im Fokus stehen hier die Instruktion und das Verstehen der Roboter Fähigkeiten bezüglich gegebener Aufgaben.

Kapitel 3 stellt eine neue Methode vor, die ungelabelte Demonstrationen in Sequenzen von Bewegungsprimitiven segmentiert und gleichzeitig eine Bibliothek von Primitiven lernt. Diese Methode erlaubt es einem Nicht-Experten eine gesamte Aufgabe zu demonstrieren anstelle jedes einzelnen Primitives. Bewegungsprimitive repräsentieren eine einfache, atomare und üblicherweise parametrisierte Bewegung. Die präsentierte Methode segmentiert jede Demonstration, indem sie ähnliche, wieder auftretende Muster über alle Demonstrationen hinweg identifiziert und diese als Zufallsvariable einer stochastischen Repräsentation eines Bewegungsprimitivs behandelt. Die Methode ist als ein Expectation-Maximization Ansatz formuliert und wurde in mehreren Aufgaben evaluiert, inklusive dem Zusammenbau eines einfachen Stuhls und das Segmentieren von Tischtennis Demonstrationen.

In Kapitel 4 werden die zuvor segmentierten Demonstrationen und die gelernten Primitive dazu genutzt, formale Grammatiken für Bewegungsprimitive zu lernen. Formale Grammatiken sind ein etabliertes Konzept zur Analyse formaler Sprachen und wurden in diversen Gebieten angewendet, beginnend in der Linguistik, über

Compiler Architektur bis hin zur Robotik. Die einfachste Klasse der Grammatiken, reguläre Grammatiken, stimmen in ihrer stochastischen Form mit Hidden Markov Modellen überein. Die intuitive, hierarchische Repräsentation der Transitionen als Regeln hingegen, vereinfacht es Nicht-Experten das von der Grammatik beschriebene Verhalten zu verstehen. Eine Sequenz von Bewegungen kann nun als ein von einer Grammatik produzierter Satz betrachtet werden. Die Produktion jedes Satzes kann durch eine Baumstruktur, Parse Tree, dargestellt werden, was ein einfaches Verstehen der beteiligten Regeln erlaubt. Stochastische kontextfreie Grammatiken sind eine Übermenge der regulären Grammatiken und sind daher ebenfalls ausdrucksstärker als Hidden Markov Modelle. Das Lernen von stochastischen kontextfreien Grammatiken für natürliche Sprachen gilt als schweres, ungelöstes Problem. Allerdings, zeigen Sequenzen von Bewegungsprimitiven wesentlich einfachere Strukturen auf, die das Lernen von Grammatiken erleichtern. Die präsentierte Methode wurde erfolgreich auf verschiedenen Aufgaben evaluiert, wie zum Beispiel das Platzieren von Spielsteinen in einem Tic-Tac-Toe Szenario oder das Übergeben von Teilen beim Zusammenbauen einer Holzkiste.

Kapitel 5 führt das Konzept des Reinforcement Learning in das Gebiet der formalen Grammatiken ein. Wir wenden den Natural Policy Gradient Ansatz an um die Grammatik Parameter zu lernen die Sequenzen produzieren die wiederum eine gegebene Ausgabe lösen. Dies erlaubt das selbständige verbessern des Roboter-Verhaltens. Zum Beispiel, eine Aufräum-Aufgabe kann auf Effizienz optimiert werden, während Selbstkollisionen des Roboters vermieden werden. Die Grammatik Parameter entsprechen den Wahrscheinlichkeiten der Produktionen einer jeden Regel. Daher muss sicher gestellt werden dass die Wahrscheinlichkeits Bedingungen eingehalten werden, während die Parameter gelernt werden. Die angewendete Natural Policy Gradient Methode garantiert genügend kleine Parameter Anpassungen, dass die Parameter graduell angepasst werden. Wir leiten die Natural Policy Gradient Methode für Formale Grammatiken her und evaluieren sie auf mehreren Aufgaben.

Zusammen bilden die in dieser Dissertation vorgestellten Ansätze eine Imitation Learning Pipeline die das Instruieren, Interagieren und Kollaborieren mit Robotern vereinfacht. Ausgehend von ungelabelten Demonstration wird eine Bibliothek von Bewegungsprimitiven gelernt, während die Demonstrationen zeitgleich in Sequenzen von Primitiven segmentiert werden. Diese Sequenzen werden dann genutzt um eine Formale Grammatik für Bewegungsprimitive zu lernen. Die Struktur der Grammatik und die entsprechenden Parse Trees formen eine verständliche Repräsentation der Roboter-Fähigkeiten bezüglich einer gegebenen Aufgabe. Abschließend ermöglicht ein Reinforcement Learning Ansatz die selbstständige Optimierung der Grammatik hinsichtlich einer gegebenen Zielfunktion.

Acknowledgements

I have received a tremendous amount of support over the past five years from fellow researchers, family and friends without which this thesis would not have been possible. I would therefore like to thank:

- Jan Peters for all the support, advice and knowledge he shared with me over the years. Thanks to his guidance and his trust in me, I became the researcher I am today.
- Gerhard Neumann for suggesting a research career to me, for being an always available, constant source of knowledge and for becoming a trusted and valued friend.
- Guilherme Maeda for guiding me through my Ph.D. and for never forcing a direction on me but instead supporting my decision and helping me to reflect on them.
- my thesis referees Prof. Jan Peters and Prof. Ken Goldberg for evaluating this thesis.
- Prof. Andreas Koch for heading the thesis committee and Prof. Oskar von Stryk and Prof. Kristian Kersting for participating in the defence.
- all my co-authors for many great collaborations, and in particular Jan Peters, Guilherme Maeda, Gerhard Neumann, Filipe Veiga, Kristian Kersting, Oliver Kroemer, Alexandros Paraschos, Marco Ewerton, Takayuki Osa, Dorothea Koert, Daniel Wilbers and Abbas Abdolmaleki.
- Elmar and Tucker for teaching me to extract valuable information from negative feedback. You will always be Reviewer 2 to me.
- Oliver for setting up and teaching me about DarIAS and Alexandros for enhancing SL and always helping with questions and improvements.
- Alexandros, Roberto, Herke, Christian and Oliver for listening, helping, teaching, criticizing, teasing and just being all around great big Ph.D. brothers.
- Boris and Hany for many interesting discussion and for listening to my complaints and my bad puns.
- my fellow colleagues that made my time at IAS an unforgettable experience, in particular Marco, Gregor, Fabio, Samuele, Dorothea, Svenja, Daniel, Simone, Oleg, Michael, Joni and Riad.
- Filipe for always listening to ideas and giving me support and feedback, for listening to my complaints and for always having my back, for the countless evenings we ate, drank and laughed together. We started our Ph.D.'s together as colleagues but we finished them together as brothers.
- Evelyn and Denise for being the best flatmates I could have possibly hoped for.
- Sebastian, Jessica, Mathias and the entire BFG for being amazing friends that always made sure that I had enough distraction when I needed it.
- my parents Inga and Dmitrii for their unconditional love, support and believe in me. Everything I am and ever will be I owe to them.
- my siblings Anatoli and Michelle for sparking and nurturing the desire to learn and to teach in me.
- my wonderful wife Lisa for supporting me, motivating me, picking me up, protecting me and for being in my life and making it exponentially better in every possible way.



Contents

Abstract	iii
Zusammenfassung	v
Acknowledgements	vii
1. Introduction	1
1.1. Contributions	1
1.2. The Imitation Learning Pipeline	3
2. Movement Primitives and Grammars	5
2.1. Probabilistic Movement Primitives	5
2.2. Formal Grammars	6
3. Learning Movement Primitive Libraries through Probabilistic Segmentation	11
3.1. Problem Statement and Notation	13
3.2. Related Work	15
3.3. Learning Movement Primitive Libraries via Probabilistic Segmentation .	16
3.3.1. Probabilistic Inference on Segmentations	17
3.4. Experimental Evaluation of ProbS	25
3.4.1. Setup of the Real Robot Writing Task	26
3.4.2. Setup of the Real Robot Chair Assembly Task	32
3.4.3. Robot Table Tennis	34
3.5. Conclusion	35
4. Learning Attribute Grammars from Movement Primitives Sequences	37
4.1. Related Work	39
4.2. Problem Statement	40
4.3. Identifying the Primitive Category	42
4.4. Determining Connectibility of Primitives	44
4.5. Inducing PCFGs for Movement Primitives	45
4.5.1. Learning Grammars through Posterior Optimization	46
4.5.2. Traversing the Grammar Space \mathcal{G}	48
4.5.3. Finding \mathcal{G}^*	50
4.6. Enhancing PCFGs with Attributes for Movement Primitive Sequencing .	52
4.6.1. Evaluation Scheme for the Tic-Tac-Toe Task	53
4.6.2. A General Evaluation Scheme for Sequencing Tasks	55
4.6.3. Evaluating Parallel Attribute Sets	56

4.7. Experiments	57
4.7.1. Learning a Grammar for Tic-Tac-Toe Turns	57
4.7.2. Learning a Grammar for a Simple Toolbox Assembly	60
4.8. Conclusion	62
5. Reinforcement Learning for Formal Grammars	65
5.1. Natural Policy Gradient for Formal Grammars	66
5.1.1. Grammar as Policy	67
5.1.2. Natural Policy Gradient for Grammars	68
5.2. Experiments	69
5.2.1. Learning a Desired Length for a^nb^n	69
5.2.2. Learning to Play Tic-Tac-Toe	71
5.2.3. Learning to Clean the Tic-Tac-Toe Board	74
5.3. Conclusion	78
6. Conclusion	79
6.1. Summary	79
6.2. Future Work	80
6.3. Lessons Learned	82
A. Publication List	83
A.1. Journal Articles	83
A.2. Articles in Conference Proceedings	84
A.3. Workshop Papers	85
B. Curriculum Vitæ	87
Bibliography	97

Notation

The following list introduces the basic notation used throughout this thesis. Due to the vast amount of variables the symbols are defined in the chapters in which they are used. In the case that some symbols are overloaded across chapters, the correct meaning will be apparent from the context and the respective chapter.

Notation	Description
x	scalar
\mathbf{x}	column vector
\mathbf{x}^T	transposed column vector, i.e., row vector
x_i	i^{th} element of the vector \mathbf{x} if there is a vector \mathbf{x}
\mathbf{x}_i	column vector indexed by i . For instance, the i^{th} sample of a multivariate distribution
$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n]$	matrix consisting of column vectors $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n$
$\mathbf{X}^T = [\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n]$	transposed matrix
$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n\}$	set of elements $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n$
$\mathcal{X} = \langle \mathcal{A}_1, \mathcal{A}_2, \dots \mathcal{A}_n \rangle$	n -tuple with elements $\mathcal{A}_1, \mathcal{A}_2, \dots \mathcal{A}_n$
a	terminal symbol in the context of formal grammars
A	nonterminal symbol in the context of formal grammars
$\text{func}(\cdot)$	defined function
$p(\mathbf{x})$	distribution over \mathbf{x}
$p(\mathbf{x} \boldsymbol{\theta})$	distribution over \mathbf{x} given $\boldsymbol{\theta}$



1 Introduction

One of the declared goals of robotics is the introduction of robots as servants, helpers and coworkers. In the recent past, technological advances in robotics as well as machine learning allowed robots and autonomous agents to take on an ever bigger role in our day to day live and at work. The increasing complexity of the robotic systems allows them to physically solve complicated tasks in human-oriented environments. In the not too far future robots in private households will be considered as natural as computers. In order to persist in these scenarios, they need to learn new tasks and adapt existing ones. This modifications have to happen in a, for a non-expert, intuitive way.

Currently, however, the programming and understanding of such complex systems is reserved to a few experts. This is very clearly reflected in the current deployment of robots in the large scale manufacturing industry. Robots are programmed once by an expensive expert to repeat the same task several million times. Changes in the executed task require an modification by the expert, and therefore become very expensive, very quickly. Hence, the deployment of robots in small and medium sized enterprises is too costly, given that smaller quantities and additional customization would require frequent modifications. Further, the environments and the required tasks in private households are more dynamic by nature. In this thesis we present a pipeline of several methods, as illustrated in Figure 1.1, that allow an intuitive teaching of new tasks and a easily comprehensible representation of the robots capabilities with respect to the taught tasks, in the form of formal grammars. In particular, the thesis focuses on probabilistic context-free grammars and attribute grammars. The choice to encode the manipulative capabilities of the robot through a concept developed to formalize natural languages is inspired by observations made in child development studies. Several studies have shown a significant relationship between motor skill and language development in infants(Leonard and Hill, 2014). While a causal link between motor skill and language development is unlikely (Lenneberg, 1967), a mutual influence and an underlying cause are supported by empirical evidence (Thelen, 1992). These observations motivated the use of formal grammars to encode the behavior of robots, implying a similar underlying structure for movement and language patterns.

1.1 Contributions

The contributions of this thesis can be separated into three main chapters. Each chapter introduces an approach aimed to improve a different aspect of human robot collaboration. Chapter 3 simplifies the demonstration of entire tasks to the robot, by eliminating the need to manually define the primitives involved in the task. The grammar induction approach for movement primitives presented in Chapter 4 proposes a well studied concept for the description of formal language to the domain

of movement primitive sequencing. The hierarchical, recursive yet comprehensible structure of grammars introduces a promising description of the robots capabilities with respect to a given task. In order to improve the learned behavior further, Chapter 5 introduces the concept of reinforcement learning to formal grammars. Given a reward function, the formulated natural policy gradient method learns the grammar parameters that optimize the given objective. This allows the robot to autonomously improve on a given task with minimal necessary interference of the human. Next we give more detailed summaries of each chapter and, hence, each contribution before presenting the outline of this thesis and the resulting imitation learning pipeline.

Learning Movement Primitive Libraries via Probabilistic Segmentation

Movement primitives are a well established approach for encoding and executing movements. While the primitives themselves have been extensively researched, the concept of movement primitive libraries has not received similar attention. Libraries of movement primitives represent the skill set of an agent. Primitives can be queried and sequenced in order to solve specific tasks. The goal of this work is to segment unlabeled demonstrations into a representative set of primitives. Our proposed method differs from current approaches by taking advantage of the often neglected, mutual dependencies between the segments contained in the demonstrations and the primitives to be encoded. By exploiting this mutual dependency, we show that we can improve both the segmentation and the movement primitive library. Based on probabilistic inference our novel approach segments the demonstrations while learning a probabilistic representation of movement primitives. We demonstrate our method on two real robot applications. First, the robot segments sequences of different letters into a library, explaining the observed trajectories. Second, the robot segments demonstrations of a chair assembly task into a movement primitive library. The library is subsequently used to assemble the chair in an order not present in the demonstrations.

Learning Attribute Grammars for Movement Primitive Sequencing

Composing primitives out of an existing library has shown to be a challenging problem. We propose the use of probabilistic context-free grammars to sequence a series of primitives to generate complex robot policies from a given library of primitives. The rule-based nature of formal grammars allows an intuitive encoding of hierarchically and recursively structured tasks. This hierarchical concept strongly connects with the way robot policies can be learned, organized, and re-used. However, the induction of context-free grammars has proven to be a complicated and yet unsolved challenge. We exploit the physical nature of robot movement primitives to restrict and efficiently search the grammar space. The grammar is learned by applying Markov chain Monte Carlo optimization over the posteriors of the grammars given the observations. The proposal distribution is defined as a mixture over the probabilities of the operators connecting the search space. Moreover, we present an approach for the categorization of probabilistic movement primitives and discuss how the connectibility of two

primitives can be determined. These characteristics in combination with restrictions to the operators guarantee continuous sequences while reducing the grammar space. Additionally, a set of attributes and conditions is introduced that enhance probabilistic context-free grammar for primitive sequencing tasks with the capability to adapt single primitives within the sequence. The method was validated on tasks that require the generation of complex sequences consisting of simple movement primitives using a 7 degree-of-freedom lightweight robotic arm.

Reinforcement Learning for Formal Grammars

Formal grammars have been applied and studied in a great variety of different fields, reaching from linguistics over biology to compiler architecture, natural language processing and even robotics. However, mostly defined grammars are applied as tools for the recognition and verification of patterns rather than being used as generative models. Hence, the grammar parameters are commonly learned from batch with the goal to best explain a given amount of observed data. In this work, we introduce the concept of reinforcement learning to formal grammars, by defining and applying the natural policy gradient method to grammars. The goal is to learn parameters that find a solution to a task given a reward function. The policy is defined as the grammar itself, and the actions are represented by the sampled parse trees and hence, the sentences formed by the leaves of each tree. We define all required entities in detail and evaluate the gradient method on several learning tasks, where the objectives are to learn the parametrization of a formal grammar that maximize a given reward.

1.2 The Imitation Learning Pipeline

Each of the methods described in the three main chapters contributes to the greater framework illustrated as imitation learning pipeline in Figure 1.1. The first layer represents the required user interactions, which consist of a set of unlabeled, initial demonstrations and a kind of feedback for instance in form of a reward function. The second layer shows the methods introduced in this thesis. The Probabilistic Segmentation method, described in Chapter 3, takes unlabeled demonstrations from the user and produces a corresponding segmented set of demonstrations, while simultaneously learning a library of movement primitives. A new task can now be instructed by a non-expert via kinesthetic teaching or through observations, without the need to manually determine where one segment ends, the next one starts and which segment belongs to which primitive.

Both, the library and the segmented demonstrations, are then used to induce a probabilistic context-free grammar for movement primitives, as presented in Chapter 4. The grammar is then enriched with an attribute scheme general to movement primitive sequencing tasks. Such a grammar contains an easily understandable set of rules, reflecting the possible behavior, i.e., the possible sequences of primitives. Moreover, this grammar serves two additional purposes. Firstly, it can produce a sentence, i.e., a sequence of primitives, describing the intended behavior of the robot including a parse tree that identifies which rule was applied at which point in order to

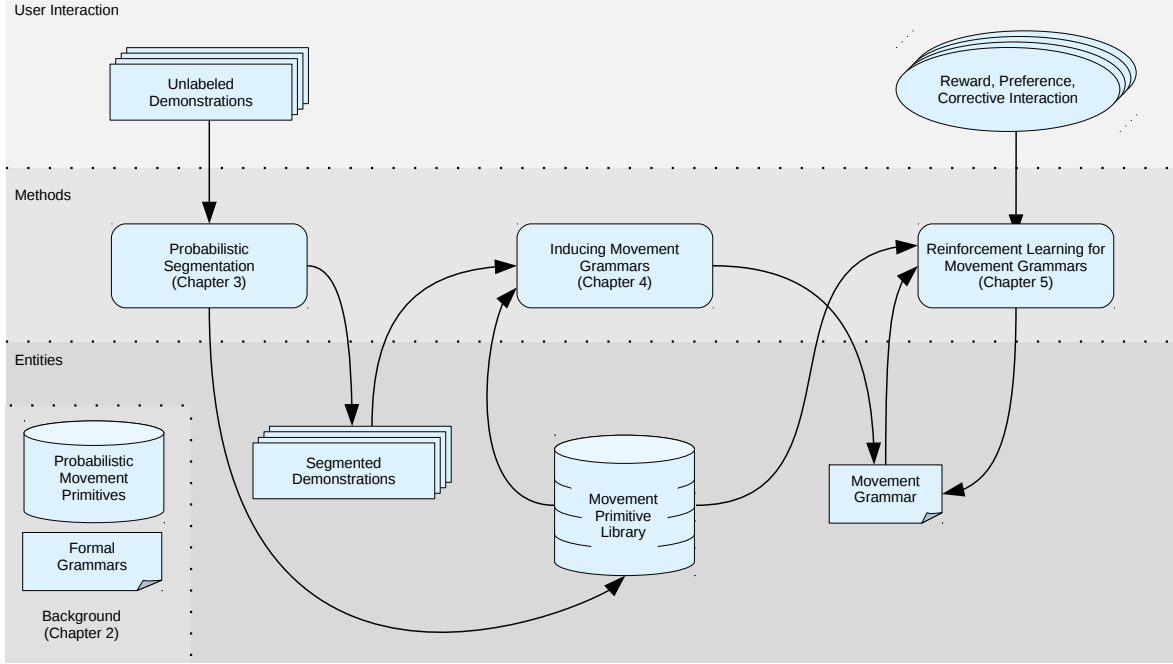


Figure 1.1.: The imitation learning pipeline. The pipeline consists of the three methods presented in this thesis. The Probabilistic Segmentation approach introduced in Chapter 3, the grammar induction presented in 4 and the natural policy gradient method for formal grammars derived in Chapter 5. Chapter 2 gives a short introduction to probabilistic movement primitives and formal grammars.

produce the sentence. The sentence informs the non-expert what the robot is about to do, providing the chance for preemptive interference. At the same time the parse tree provides an insight into why this particular sentence was produced, allowing a basic form of analysis and debugging. Secondly, the grammar can be queried to identify if a certain sentence could be produced by the contained rules, allowing to check for capability and erroneous behavior. Given a desired behavior the robot can inform the human, if a corresponding sequence of primitives is producible by the grammar and if so, with what certainty.

Finally, Chapter 5 presents how the grammar can be improved with respect to a given objective. In particular we define the learned policies as formal grammars and formulate the natural policy gradient method accordingly (Lioutikov and Peters, 2018). This approach allows the robot to improve his behavior, represented as a formal grammars, autonomously in order to achieve a goal given as a reward function.

The third layer highlights the entities produced by these methods, such as the segmented demonstrations, a library of movement primitives and the formal grammar encoding the learned robot behavior. Before each of the three methods is presented in detail Chapter 2 gives short introductions into probabilistic movement primitives and formal grammars. These introductions only cover the basics necessary to follow the contributions of this thesis.

2 Movement Primitives and Grammars

In this chapter we give short introduction into movement primitives and formal grammars. These introductions cover the necessary basic for the following chapters. A Movement Primitive (MP) encapsulates a movement or action as a discrete entity. Although simple point attractors are also sometimes referred to as MPs, MPs usually represent the shape of the movement in addition to a start and goal position. Furthermore, MPs are commonly parameterized, allowing for the modification of a MP originally learned from demonstrations. Two examples of parameterized MPs are the well known Dynamical Movement Primitives (DMPs) (Ijspeert et al., 2013a) and the more recent Probabilistic Movement Primitives (ProMPs) (Paraschos et al., 2017a). The methods presented in this thesis apply ProMPs. However, often other primitives representations such as DMPs can be used with little additional effort. The necessary adaptations and requirements for other primitive representations will be pointed out at the respective chapters. In terms of formal grammars this thesis focuses on probabilistic context-free grammars and attribute grammars. The induction method presented in Chapter 4 is not necessarily suitable for grammars that are located higher up the Chomsky hierarchy (Chomsky, 1956), such as context-sensitive grammars and unrestricted grammars. However, the reinforcement learning approach introduced in chapter 5 can be applied to any formal grammars, since it solely learns the grammar parameters and is independent of additional restrictions to the grammar rules and structure.

2.1 Probabilistic Movement Primitives

Probabilistic Movement Primitives (ProMPs) can be separated into two parts, the trajectory generating model and the variable stiffness controller. The model is represented as a Gaussian distribution over a lower dimensional projection of an observed state space trajectory. The controller projects the mean of this distribution back into the state space and tracks the new mean, while the stiffness of the tracking is determined by the covariance of the model. This methods presented in this thesis operate solely on the trajectory generating model and, hence, this introduction is limited to the relevant aspects of the model. Each observed trajectory τ is assumed to be sampled from the conditional distribution

$$p(\tau | w) = \prod_t \mathcal{N}(\tau_t | \Phi_t w, \Sigma_{\text{obs}}), \quad (2.1)$$

with Σ_{obs} being the observation noise. The feature matrix Φ_t projects the the trajectory τ onto a lower dimensional weight vector w for every time step t as defined in (Paraschos et al., 2017a). The features Φ_t are usually represented as radial basis functions for stroke like movements and von Mises functions for rhythmic move-

ments. The weight \mathbf{w} can be learned from the observed demonstration $\boldsymbol{\tau}$ by applying a maximum-a-posteriori optimization

$$\mathbf{w} = \arg \max_{\mathbf{w}'} p(\boldsymbol{\tau} | \mathbf{w}') p(\mathbf{w}'), \quad (2.2)$$

with $p(\mathbf{w}')$ denoting a prior over the weights. Depending on the prior choice the optimization yields different types of regressions. For instance, choosing a standard normal prior $p(\mathbf{w}') = \mathcal{N}(\mathbf{w}' | 0, \mathbf{I})$ yields the common ridge regression

$$\mathbf{w} = (\Phi \Phi^T + \epsilon \mathbf{I})^{-1} \Phi \boldsymbol{\tau}, \quad (2.3)$$

where $\boldsymbol{\tau}$ and Φ represent the trajectory and the features for all time steps. The features solely depend on the duration $|\boldsymbol{\tau}|$, i.e., the number of time steps in $\boldsymbol{\tau}$, and therefore, render the projected trajectory \mathbf{w} invariant to the duration of the trajectory itself. This invariance is in particular interesting, since it allows to compare trajectory of different durations purely on their shape. The projection of the demonstrated trajectories into lower dimensional spaces is a common property of movement primitives, and can, for instance, also be found in DMPs (Ijspeert et al., 2013a). However, ProMPs additionally define a Gaussian distribution over the projected trajectories

$$\mathbf{w} \sim \mathcal{N}(\mathbf{w} | \boldsymbol{\theta}), \boldsymbol{\theta} = (\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w), \quad (2.4)$$

with mean $\boldsymbol{\mu}_w$ and covariance matrix $\boldsymbol{\Sigma}_w$. Every primitive is characterized through its parameters $\boldsymbol{\theta}$ and the corresponding state space distribution is obtained by integrating out the weights

$$p(\boldsymbol{\tau} | \boldsymbol{\theta}) = \int_{\mathbf{w}} p(\boldsymbol{\tau} | \mathbf{w}) \mathcal{N}(\mathbf{w} | \boldsymbol{\theta}) d\mathbf{w}, \quad (2.5)$$

$$= \prod_t \mathcal{N}(\boldsymbol{\tau} | \Phi_t \boldsymbol{\mu}_w, \Phi_t \boldsymbol{\Sigma}_w \Phi_t^T + \boldsymbol{\Sigma}_{\text{obs}}). \quad (2.6)$$

2.2 Formal Grammars

A formal grammar is a description of a formal language in terms of symbols and production rules. The symbols, $(\mathcal{A} \cup \mathcal{V})$, are commonly separated into two disjoint sets called terminals \mathcal{A} and nonterminals \mathcal{V} , with the convention that terminals represent the atomic elements of the language, while nonterminals can be substituted with sequences of symbols. Each production rule $r_\beta \in R_\alpha$ substitutes the symbol sequence $\alpha \in (\mathcal{A} \cup \mathcal{V})^+$ with $\beta \in (\mathcal{A} \cup \mathcal{V})^+$. A rule is commonly denoted as $\alpha \rightarrow \beta$, where α and β are referred to as the left and right-hand side respectively. With these definitions a grammar can be described as a 4 tuple $\mathcal{G} = \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle$, where \mathcal{R} denotes the set of all R_α 's and \mathcal{S} is the set of all starting symbols $\mathcal{S} \subseteq (\mathcal{A} \cup \mathcal{V})^+$. A grammar can contain multiple rules for the same left hand side, i.e., $|R_\alpha| > 1$, resulting in a non-deterministic grammar. Weighting each $R_\alpha \in \mathcal{R}$ with a corresponding multinomial $\boldsymbol{\rho}_\alpha \in \Delta^{|R_\alpha|-1}$, leads to a stochastic or probabilistic grammar. Grammars can also be recursive, i.e., a series of productions starting with a rule in R_α results in a sequence containing α . The nondeterministic and recursive properties allow grammars to represent complex, hierarchical relations between symbols in relatively simply structured production rules.

$$\mathcal{G} = \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle$$

$$\mathcal{A} = \{a, b\}, \mathcal{V} = \{A, B\},$$

$$\mathcal{S} = \{A\},$$

$$\mathcal{R} = \{$$

START	→	A	(1.0)
A	→	aA	(0.3)
		aB	(0.7)
B	→	bB	(0.3)
		b	(0.7)

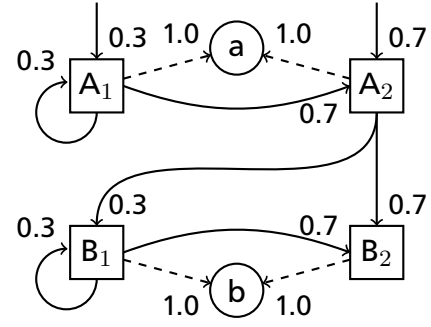
$$\}$$


Figure 2.1.: Left: A regular grammar describing the language a^+b^+ . The corresponding parse tree for the sequence $aabb$ is shown in Figure 2.2b. **Right:** A hidden Markov model equivalent to the regular grammar. Squares describe hidden states. Circles describe the emissions. Figure 2.2a shows an instantiated HMM for the sequence $aabb$.

Regular Grammar

Formal grammars are commonly classified via the Chomsky hierarchy (Chomsky, 1956). The most constrained and, therefore, least expressive grammars in this hierarchy are the so called regular grammars. Languages described by these grammars are known as regular expressions, e.g., the expression a^+b^+ represents all sentences that consist of one or more a s followed by one or more b s. Probabilistic regular grammars are equivalent to hidden Markov models (HMM). Similar to instantiated HMMs formal grammars explain observed sequences in so called parse trees. Every sentence that is part of the language described by the grammar is explained by at least one parse tree if the grammar is ambiguous and exactly one parse tree if the grammar is unambiguous. A parse tree describes the entire production of a sentence, where the leaves form the sentence and the nodes represent the nonterminals involved in the derivation of the sentence. Figure 2.1 shows a regular grammar describing the expression a^+b^+ as well as a corresponding HMM, while Figure 2.2 shows an (a) instantiated HMM and (b) a parse tree for the sequence $aabb$ of the HMM and the regular grammar respectively. The left hand side of each rule is a single nonterminal and the right-hand side can be either a terminal or a terminal followed by a nonterminal. In addition, each production of a single rule is weighted by a probability, e.g., the nonterminal A produces the sequences aA and aB with a probability of 0.3 and 0.7 respectively. While the parse tree and the graph for the time series look fairly similar we argue that the description of the higher level policy as a grammar is more intuitive to understand than the graphical model and is, therefore, better suited for non-expert users. However, given that HMMs are an extensively studied tool for the learning and analysis of time series, they are a much more common choice for the sequencing of discrete actions, such as movement primitives, than regular grammars. Despite their simplicity regular grammars can describe complex languages. In fact, every finite language is regular and can, therefore, be described by a regular grammar. Furthermore, every infinite language satisfying the pumping lemma is also regular, as for instance the described language a^+b^+ .

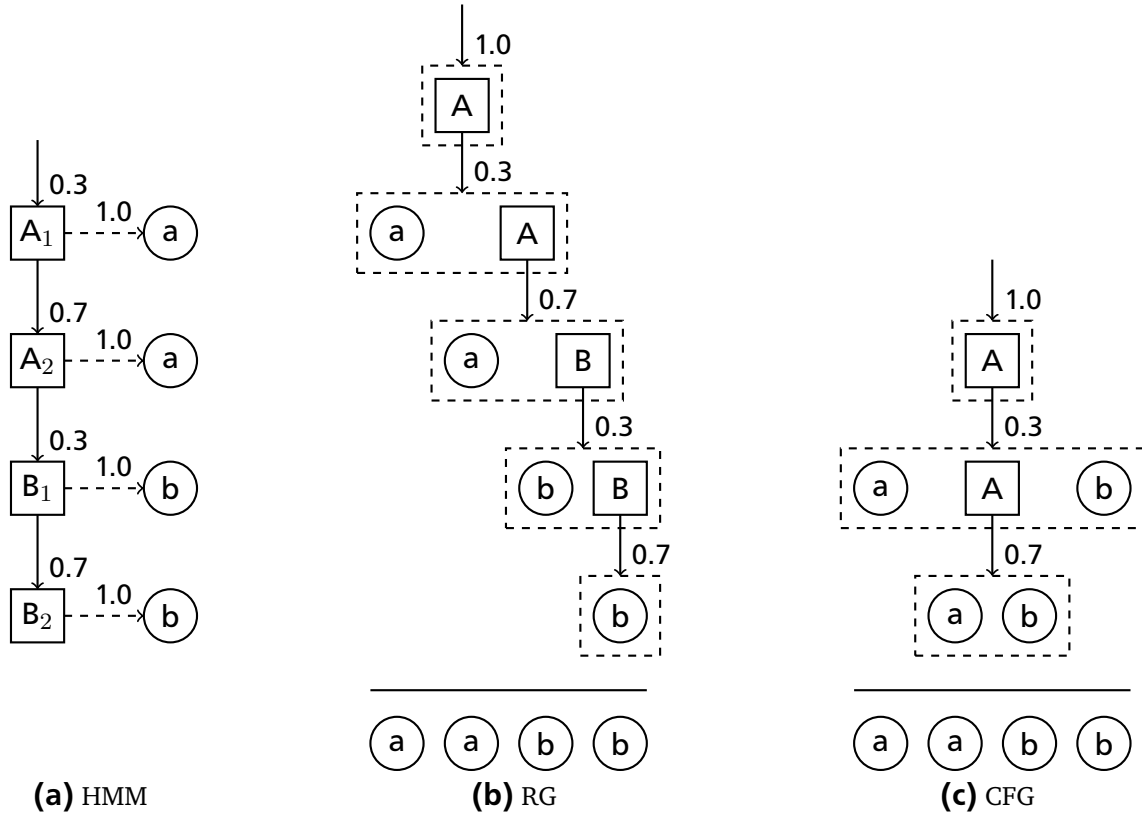


Figure 2.2.: Parse trees for the sequence aabb of (a) and (b) the hidden Markov model and the regular grammar shown in Figure 2.1. (c) shows the corresponding parse tree for the context-free grammar shown in Grammar 2.1.

Probabilistic Context-Free Grammars

Context-free grammars are able to describe infinite languages that can not be described by regular grammars, and, hence, neither by HMMs. An example for such a language is $a^n b^n$, which contains any sequence with n as that are always followed by the same number of bs. Regular grammars do not contain any mechanism to keep track of the number of produced as. Such languages can, however, be described by context-free grammars, such as

$$\begin{aligned}
 \mathcal{G} &= \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle \\
 \mathcal{A} &= \{a, b\}, \mathcal{V} = \{A\}, \mathcal{S} = \{A\} \\
 \mathcal{R} &= \{ \\
 &\quad \text{START} \rightarrow A \quad (1.0) \\
 &\quad A \rightarrow ab \quad (0.7) \\
 &\quad \quad \rightarrow aAb \quad (0.3) \\
 &\quad \}.
 \end{aligned}
 \tag{GR 2.1}$$

Context-free grammars (CFGs) still have only a single nonterminal on the left hand side, but can now contain an arbitrary sequence of terminals and nonterminals of the right-hand side.

Despite the more complex language, the context-free grammar is at least as intuitive as the previous regular grammar. Figure 2.2c shows the corresponding parse tree for the sequence `aabb`. The probabilistic extension of a context-free grammar, as in the given example, is a so called probabilistic or stochastic context-free grammar (PCFG). The nonterminal on the left hand side, `A`, can produce the sequences `ab` and `aAb` on the right-hand side with a probability of 0.7 and 0.3 respectively.

Attribute Grammars

Attribute Grammars are an enhancement of context-free grammars, where each terminal and nonterminal can be assigned multiple inherited or synthesized attributes. An inherited attribute belongs to a symbol on the right-hand side of a rule which obtains its value from attributes of the nonterminal of the left-hand side or other symbols on the right-hand side. A synthesized attribute is an attribute of the nonterminal on the left-hand side of a rule whose value is computed using attributes of the right-hand side symbols. Grammar 2.1 for instance can be transformed into the attribute grammar

$$\begin{array}{llll}
 \text{START} & \rightarrow & A & (1.0) \\
 A_1 & \rightarrow & ab & (0.7) \\
 & & A_1.\text{depth} = 1 & \\
 & \rightarrow & aA_2b & (0.3) \quad (\text{GR 2.2}) \\
 & & A_1.\text{depth} = A_2.\text{depth} + 1 & \\
 & & A_2.\text{max_depth} = A_1.\text{max_depth} - 1 & \\
 & & \text{assert: } A_1.\text{max_depth} > 0, &
 \end{array}$$

with `depth` and `max_depth` being attributes. The indices of `A1` and `A2` simply distinguish between the same nonterminal within a single rule. The synthesized attribute `depth` evaluates to the number of recursions that occurred during the production of a sentence, while the inherited attribute `max_depth` defines how many recursions are at most supposed to occur. The latter is achieved by defining the condition that the second rule is only chosen if `A1.max_depth > 0`, resulting in sentences with at most `max_depth` number of `as` and `bs`. Such conditions extend the expressiveness of attribute grammars beyond the one of context-free grammars. For instance, the language `an bn cn` can not be represented by context-free grammars but easily through attribute grammars.

Grammar Induction

Learning formal grammars from observed sequences of terminals is referred to as Grammar Induction. Commonly the task is formulated as a search through a grammar space \mathcal{G} , where the connections between grammars are represented as different operators. Such operators manipulate the set of production rules \mathcal{R} and the set of nonterminals \mathcal{V} accordingly. Starting from an initial grammar \mathcal{G}^0 these operators are used to traverse the grammar space, searching for the optimal grammar \mathcal{G}^* . Various search strategies have been suggested, e.g., beam search (Stolcke, 1994; Lee et al., 2013) and Markov chain Monte Carlo optimization (Talton et al., 2012).



3 Learning Movement Primitive Libraries through Probabilistic Segmentation

A key challenge of human-robot collaboration is the teaching of new tasks to the robot. Ideally, the robot is taught in an intuitive way, that does not require extensive expert knowledge. A commonly followed concept to achieve such behavior is imitation learning. The robot is provided with one or more demonstrations of a task, which the robot subsequently reproduces and improves. Often, an entire task consists of a single motion, encoded as a single movement primitive (Mülling et al., 2010),(Paraschos et al., 2013). This concept has been applied in a variety of tasks, including hitting movements in table tennis (Mülling et al., 2010) and locomotion (Nakanishi et al., 2004).

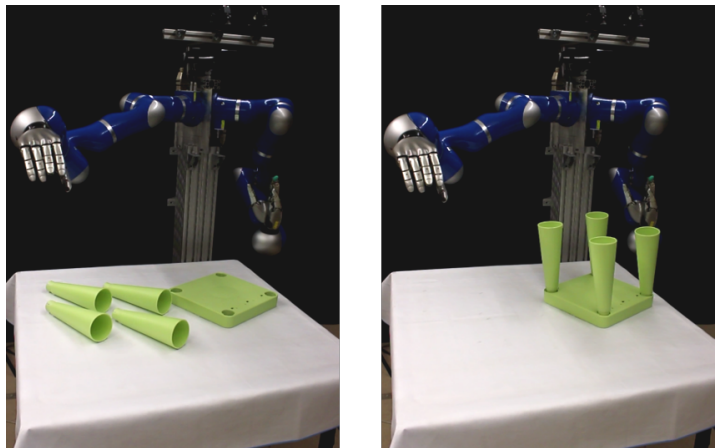


Figure 3.1.: The robot platform used for a chair assembly experiment. We used a seven DoF KUKA lightweight arm equipped with a five finger DLR HIT Hand II as end effector. The executed movement primitives were learned by segmenting human demonstrations.

Solving more complex, non-monolithic tasks with a single movement primitive may result in a great loss of generality. Considering complex tasks as a sequence of primitives offers multiple advantages. For example, primitives can be easily generalized and optimized between the points where they connect. The same set of primitives can be reused to execute different tasks, and the movement plan can be adapted by replacing one primitive within the sequence by another one. A fundamental problem of such approaches is the autonomous acquisition of these primitives without relying on hand labeled demonstrations. In this chapter, we address this problem by proposing a framework for segmenting unlabeled demonstrations into a library of movement primitives.

Essentially, such movement primitive acquisition consists of two problems, the segmentation of observed trajectories and the learning of the underlying movement primitive library. In this chapter we tackle these two problems in conjunction. Each demonstrated trajectory can be considered a multidimensional time series. A common way to segment time series data is to apply heuristics. However, the quality

of such heuristics and therefore, the corresponding segmentation is often task dependent. For instance, while an assembly task consisting of point to point motions might be well segmented at zero crossing velocities, the same heuristic applied on continuously written words might achieve poor, meaningless results. Furthermore, different parts of the data could be best explained by different heuristics, which raises the problem of identifying at what point to apply which heuristic.

Our approach starts from the premise that a task-specific heuristic can only segment a given trajectory sub-optimally, therefore, leading to a low-quality library. As a consequence, some movement primitives may not be meaningful while others will suitably describe the data. Our method applies probabilistic inference to reason iteratively over all possible segmentations by learning a probabilistic representation of movement primitives from a weighted set of segments. In return, the learned primitives are used to improve the set of segments by down-weighting segments that are less plausible given the current movement primitive library. We provide the mathematical formulation for the solution of this problem as an iterative Expectation-Maximization (EM) algorithm and show that our algorithm converges to a compact set of movement primitives given over-segmented demonstrations.

Another interesting aspect is the relationship between the size of the library and the complexity of the contained primitives. The learned primitives should be complex enough to represent a dedicated motion, while being simple enough to qualify as a modular unit. The size of the library is not directly proportional to the complexity of the contained primitives. The most complex primitive is the one learned from an entire demonstration. This choice would lead to a library the same size as the number of demonstrations. Simpler primitives that are shared by multiple demonstration, however, can result in a more compact library. A useful metric, that we apply in this chapter, is the bit-encoding of the observed demonstrations. The encoding is described in more detail in the experiment section.

In summary, the main contribution of this chapter is the Probabilistic Segmentation (ProbS) algorithm that concurrently improves a given segmentation and the library of movement primitives. The method was validated on a real robot platform. We evaluated and compared our method on a letter segmentation task. The robot was taught different sequences of letters and subsequently executed the primitives learned by ProbS. Additionally, we applied ProbS to a chair assembly task. The required movement primitive library was learned by segmenting the human motion of a chair assembly. Subsequently, movement primitives were sequenced from the learned library to assemble the chair in a previously undemonstrated order.

The chapter is organized as follows. First the problem statement and the used notation is given. Next, related work is presented and discussed. Followed by the introduction of the Probabilistic Segmentation approach. Afterwards, we compare the proposed method to a baseline method and a state-of-the-art segmentation method, using a writing and an assembly task. Both tasks were performed on a real robot platform and show the capabilities of our method. Additionally, we use the method to segment table tennis strokes from kinesthetic teaching.

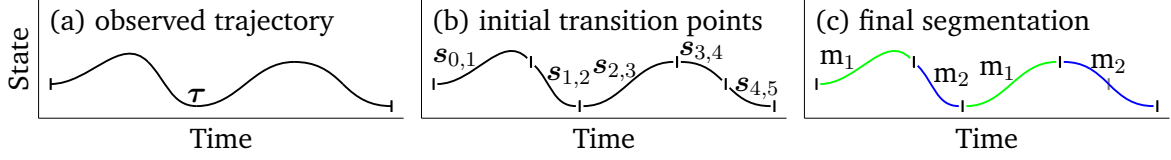


Figure 3.2.: An illustration of a possible segmentation. (a) shows a one dimensional, continuous observation. (b) shows four initially suggested transitions, illustrated as black bars, and the five resulting segments, if all transitions were true positives. (c) shows a possible segmentation. The fourth transition was identified as a false positive transition, illustrated as a gray bar. Two primitives m_1 and m_2 were learned from the resulting four segments.

3.1 Problem Statement and Notation

Given a set of observed trajectories $\mathcal{T} = \{\tau_1, \dots, \tau_{|\mathcal{T}|}\}$, the goal of this work is to learn a set of underlying movement primitives $\mathcal{M} = \{m_1, \dots, m_{|\mathcal{M}|}\}$ which explains \mathcal{T} , i.e., the movement primitive library which produced \mathcal{T} . Since we are interested in the underlying library \mathcal{M} of a task domain, the trajectories in \mathcal{T} can describe the same or multiple tasks, as long as they belong to the same domain, and, therefore, can be explained by the same library. The duration of each individual demonstration τ might be different. An example of a one dimensional trajectory is illustrated in Figure 3.2a. For each trajectory $\tau \in \mathcal{T}$, a set of possible transition points \mathcal{C}_τ is defined. Each transition point $h, i, j \in \mathcal{C}_\tau$ separates two subsequent segments, e.g., $s_{h,i}$ and $s_{i,j}$. The indices of $s_{i,j}$ denote that the segment starts at the i^{th} and ends at the j^{th} transition point. We will drop the indices whenever it is irrelevant at which transition point the segment starts $s := s_{i,j}$.

The set \mathcal{C}_τ splits τ into a set of segments \mathcal{S}_τ , which represents a possible segmentation of τ . We assume that \mathcal{C}_τ over-segments τ , i.e., the transition points producing the correct segmentation \mathcal{S}_τ^* are a subset of the initial transition points $\mathcal{C}_\tau^* \subseteq \mathcal{C}_\tau$. The transition points included in the correct set of transitions \mathcal{C}_τ^* are referred to as true positive transitions and, respectively, the transition points not included, i.e., $\mathcal{C}_\tau \setminus \mathcal{C}_\tau^*$ are referred to as false positive transitions.

Unfortunately, the true positive transitions are unknown. Therefore, every possible subset $\mathcal{C}'_\tau \subseteq \mathcal{C}_\tau$ has to be considered. Each \mathcal{C}'_τ results in a different segmentation. The set of all possible segmentations will be denoted as $\mathcal{S}_\tau \in \mathcal{D}_\tau$. Furthermore, some segments occur in multiple segmentations, therefore, the set of all possible segments is given as $\mathcal{A}_\tau = \bigcup_{\mathcal{S}_\tau \in \mathcal{D}_\tau} \mathcal{S}_\tau$ and the set of all segmentations containing a particular segment s is defined as $\mathcal{D}_s = \{\mathcal{S} | s \in \mathcal{S}, \forall \mathcal{S} \in \mathcal{D}\}$.

Our method tackles two challenges simultaneously: eliminating all false positive transitions $\mathcal{C}_\tau \setminus \mathcal{C}_\tau^*$, therefore determining the correct segmentation $\mathcal{S}_\tau^* \in \mathcal{D}_\tau$ and learning the underlying MP library \mathcal{M} from the chosen segments $s_\tau \in \mathcal{S}_\tau^*, \forall \tau \in \mathcal{T}$. The segments $s_\tau \in \mathcal{A}_\tau$, the segmentations $\mathcal{S}_\tau \in \mathcal{D}_\tau$ and the transitions \mathcal{C}_τ are always defined with respect to a single trajectory $\tau \in \mathcal{T}$. For simplicity, we will drop the subscript τ from now on. Table 3.1 summarizes the defined entities alongside others that will be defined later in the chapter.

Symbol	Description
τ	an observed multidimensional, unlabeled trajectory
\mathcal{T}	a set of trajectories $\mathcal{T} = \{\tau_1, \dots, \tau_{ \mathcal{T} }\}$
m_k	a movement primitive parameterized by θ_k
θ_k	a parameterization of primitive m_k
\mathcal{M}	a set of primitives $\mathcal{M} = \{m_1, \dots, m_{ \mathcal{M} }\}$
Θ	a set of all primitive parameters, i.e., $\Theta = \{\theta_1, \dots, \theta_{ \mathcal{M} }\}$
$\mathcal{C} := \mathcal{C}_\tau$	a set of possible transition points for τ
$s := s_{i,j}$	a segment, i.e., the part of τ between transition point i and j
w	a lower dimensional projection of the segment s
\mathcal{W}	a set of all projections across all trajectories, i.e., $\mathcal{W} = \{w \mid w = v(s), \forall s \in \mathcal{A}_\tau, \forall \tau \in \mathcal{T}\}$
α_s, α_w	a segment weighting defining how likely s is part of the underlying segmentation
$\mathcal{S} := \mathcal{S}_\tau$	a possible segmentation of τ , i.e., a set of segmentations defined by a subset of \mathcal{C}_τ , e.g., $\mathcal{S}_\tau^{[1]} = \{s_{0,1}, s_{1,4}, s_{4,5}\}$, $\mathcal{S}_\tau^{[2]} = \{s_{0,5}\}$, $\mathcal{S}_\tau^{[3]} = \{s_{0,1}, s_{1,2}, s_{2,3}, s_{3,5}\}$
$\mathcal{D} := \mathcal{D}_\tau$	a set of all possible segmentations for τ , i.e., $\mathcal{D}_\tau = \{\mathcal{S}_\tau^{[1]}, \mathcal{S}_\tau^{[2]}, \mathcal{S}_\tau^{[3]}, \dots\}$
$\mathcal{D}_s^\rightarrow$	a set containing all segmentations of the partial trajectory that starts at the beginning of τ and ends at the beginning of s
\mathcal{D}_s^\leftarrow	a set containing all segmentations of the partial trajectory that starts at the end of s and ends at the end of τ
$\mathcal{A} := \mathcal{A}_\tau$	a set of all possible segments across all possible segmentations for τ , i.e., $\mathcal{A}_\tau = \bigcup_{\mathcal{S}_\tau \in \mathcal{D}_\tau} \mathcal{S}_\tau$
$\mathcal{A}^{\text{start}}$	a subset $\mathcal{A}_\tau^{\text{start}} \subseteq \mathcal{A}_\tau$ that contains all segments that start at the beginning of τ
\mathcal{A}^{end}	a subset $\mathcal{A}_\tau^{\text{end}} \subseteq \mathcal{A}_\tau$ that contains all segments that end at the end of τ
$\mathcal{A}_s^{\text{pred}}$	a subset $\mathcal{A}_s^{\text{pred}} \subset \mathcal{A}_\tau$ that only contains segments that end at the beginning of s
$\mathcal{A}_s^{\text{succ}}$	a subset $\mathcal{A}_s^{\text{succ}} \subset \mathcal{A}_\tau$ that only contains segments that begin at the end of s
$\gamma_s^\rightarrow, \delta_s^\rightarrow$	forward messages send from or towards s respectively
$\gamma_s^\leftarrow, \delta_s^\leftarrow$	backward messages send from or towards s respectively

Table 3.1.: The main entities defined across the chapter. In the chapter the subscript τ and the segment indices i, j are dropped whenever irrelevant.

3.2 Related Work

Algorithms for automatic segmentation have been investigated extensively, not only for the purposes of generation of robot movement primitives but mainly as a general tool for movement analysis and classification. Hidden Markov Models (HMMs) have been widely adopted in this context. For example, (Brand and Kettner, 2000) analyzed video images to train a HMM to classify if a person is walking, running or crouching. (Takano and Nakamura, 2006) used automatic segmentation of motion patterns based on HMMs to group segments hierarchically, where higher level representations of symbols can then be used to orchestrate and generate low level robot movements. More recently, (Kulic et al., 2009) proposed an on-line segmentation method based on HMMs that creates a tree of primitives; the lower nodes representing detailed movements with generality increasing towards the root. HMMs have also been used in conjunction with the superposition of movement primitives for the specific case of handwriting analysis, (Williams et al., 2008). In, (Krishnan et al., 2015) a Dirichlet Process Gaussian Mixture Model is learned to identify the transition points between the segments. The transition points are subsequently clustered spatially and temporally. In general, HMMs and methods that explicitly address temporal sequences, e.g., (d’Avella and Tresch, 2001), have been generally accepted for segmentation. In this chapter, however, we opt for a shape-based clustering approach on the basis that our desired library must be invariant to the possible combinations of movement primitives transitions. The encoding of trajectories that do have a sequential pattern are naturally addressed by our method as it maintains only the most probable combinations of segments.

Our work takes advantage of movement primitive representations that are time invariant, such as Dynamical Movement Primitives (DMPs), (Ijspeert et al., 2013b), or Probabilistic Movement Primitives (ProMPs), (Paraschos et al., 2013). Segmentation with movement primitives is particularly suited for library construction as segments with the same profile, but with different time scales are treated as the same primitive. (Chiappa and Peters, 2010), for example, had to take the expected time scales of possible segments into account with the introduction of a heuristic about the minimum and maximum duration of the movement primitives; in our approach such user-defined inputs are not necessary.

From the movement primitive perspective, our algorithm relates to the work of (Meier et al., 2011), and (Niekum et al., 2013) where DMPs have been used in different ways. In the first approach, a library of primitives is assumed given, while in our work we design our algorithm to start from an empty set. Compared to the work of (Niekum et al., 2013), the authors treat segmentation as an independent initial step which therefore, later affects the reconstruction of a task, in this case using finite state automata. As a consequence, interactive corrections given by a human demonstrator are introduced. In contrast, our approach treats segmentation and primitive learning as an iterative optimization process where both are intrinsically connected.

Hierarchical skills have been explored by (Takano and Nakamura, 2006), (Kulic et al., 2009), (Konidaris et al., 2012), and (Yamane et al., 2011), and can be very efficient for on-line applications or to represent different granularities in the task. The

philosophy of our method differs in the sense that we do not enrich a model by adding branches, but instead prune unnecessary segments given by a possibly erroneous initial heuristic. We leverage on batch, off-line learning to essentially reconstruct the movement primitive library, iteratively. This leads to a single level representation which decreases the number of segments as the library is improved after each EM iteration.

A general problem in movement segmentation and library generation is the trade off between the generality of the method and its tractability, the latter usually achieved by the introduction of heuristics. For example, Zero Crossing Velocity (ZCV) has usually been used as an intuitive criterion to obtain the initial segmentation of trajectories (Fod et al., 2002), (Nakazawa et al., 2002), (Barbič et al., 2004). In the context of movement primitives, however, ZCV usually leads to over-segmentation, especially when the robot moves at low speeds. (Lemme et al., 2014) proposed segmenting demonstrations based on geometric similarities. (Wächter and Asfour, 2015) introduced a two layer hierarchical approach is proposed to segment 6D motion trajectories. The top layer identifies segments based on semantic criteria, e.g., contact between objects. The bottom layer applies a heuristic to identify keyframes based on the difference of the adjacent trajectory parts. (Endres et al., 2013) apply bayesian binning in order to segment end-effector trajectories into pieces with different parameter values of the two-third power law. Other heuristics applied to segmentation include velocity profiles and minimum jerk, (Rohrer and Hogan, 2006), changes of the system dynamics, (Kroemer et al., 2014). Our work differs by being insensitive to the particular choice of the heuristic. Our assumption is that a given heuristic will lead to an initial number of excessive segments, which will be then optimized by decreasing the occurrence of transitions among them when necessary.

3.3 Learning Movement Primitive Libraries via Probabilistic Segmentation

We assume that each observed trajectory $\tau \in \mathcal{T}$ can be represented by one of the corresponding segmentations $\mathcal{S} \in \mathcal{D}$. The trajectory τ can be explained by concatenating the segments contained in \mathcal{S} .

The method is initialized with a set of possible transition points \mathcal{C} , which divides each trajectory into multiple segments as shown in Figure 3.2. We assume that \mathcal{C} weakly over-segments τ and that there is a subset of true positive transitions $\mathcal{C}^* \subseteq \mathcal{C}$ which results in the correct segmentation $\mathcal{S}^* \in \mathcal{D}$.

Our goal is to determine the correct segmentation \mathcal{S}^* while simultaneously learning the underlying library \mathcal{M} . Since \mathcal{S}^* is not known, we treat \mathcal{S}^* as a latent variable. Our proposed approach assesses the quality of all possible segmentations $\mathcal{S} \in \mathcal{D}$ by applying probabilistic inference methods to learn locally optimal \mathcal{M} and \mathcal{S} in conjunction. The size of the library $|\mathcal{M}|$, i.e., the number of learned primitives, is determined by applying a bisecting k-means algorithm on the segments $s \in \mathcal{S}$ defined by the current segmentation. Therefore, no prior knowledge about the number of primitives is expected. Furthermore, given that the segmentation itself is learned, the number of primitives change with the segmentation estimate. Figure 3.2 illustrates the segmentation process of our method. The found movement primitives are shown

in Figure 3.2c, the observed trajectories in Figure 3.2a and the transition points in Figure 3.2b.

Defining the Transition Points \mathcal{C}

The results of the proposed method depend on the set of possible transitions \mathcal{C} . Given the Expectation-Maximization like nature of the method, it is unfeasible to initialize it with a transition point for every time step of the observation. A possibility to restrict \mathcal{C} to a manageable size is to initially use heuristics to determine \mathcal{C} . These heuristics can be chosen task specifically and different heuristics can also be combined seamlessly. However, the method only considers the transitions contained in \mathcal{C} , i.e., it is restricted to eliminate false positive transitions. Therefore, \mathcal{C} has to provide a weak over-segmentation, i.e., $\mathcal{C}^* \subseteq \mathcal{C}$, where \mathcal{C}^* denotes the set of true positive transitions.

Movement Primitive Representation

We apply probabilistic movement primitives as defined in Chapter 2. However, the primitives are not defined over the entire trajectory, but rather over each individual segment $p(s | \theta_k)$. Therefore, if the segment s is a valid segment, there exists an underlying movement primitive m_k which produced the corresponding projected segment

$$w \sim \mathcal{N}(w | \mu_k, \Sigma_k).$$

Moreover, we only consider correlations between the dimensions and not between the time steps, i.e.

$$p(s_t | \theta_k) = p(s_t | \theta_k, s_{t-1}),$$

where s_t describes the segment s at the time step t , i.e., s_t is the t^{th} entry of s . Therefore, the probability of a segment s given a movement primitive m_k is defined as

$$p(s | \theta_k) = \prod_{t=1}^{|s|} p(s_t | \theta_k), \quad (3.1)$$

and, the probability for a single time step t given the movement primitive m_k is

$$p(s_t | \theta_k) = \mathcal{N}(s_t | \phi_t^T \mu_k, \phi_t^T \Sigma_k \phi_t), \quad (3.2)$$

where the feature vectors ϕ_t^T are the corresponding rows of the feature matrix Φ .

3.3.1 Probabilistic Inference on Segmentations

Each movement primitive $m \in \mathcal{M}$ is represented by a parameterized, generative model $p(s | \theta_k)$, with θ_k denoting the parameters of the k^{th} movement primitive m_k .

However, we do not consider the single primitives independently but assume that each segment was drawn from the entire library, modeled as a mixture of primitives

$$\begin{aligned} s &\sim p(s | \Theta), \\ p(s | \Theta) &= \sum_{k=1}^{|\mathcal{M}|} \lambda_k p(s | \theta_k), \\ \Theta &= \{(\lambda_1, \theta_1), \dots, (\lambda_{|\mathcal{M}|}, \theta_{|\mathcal{M}|})\}, \end{aligned} \quad (3.3)$$

where λ_k denotes the mixing coefficient for movement primitive m_k .

Furthermore, we assume that every observed trajectory $\tau \in \mathcal{T}$ was drawn from a parameterized generative model

$$\tau \sim p(\tau | \Theta, \mathcal{S}^*) = \prod_{s \in \mathcal{S}^*} p(s | \Theta). \quad (3.4)$$

Since \mathcal{S}^* is unknown, we treat it as a latent variable, and integrate it out, which leads to

$$p(\tau | \Theta) = \sum_{\mathcal{S} \in \mathcal{D}} p(\mathcal{S}) \prod_{s \in \mathcal{S}} p(s | \Theta). \quad (3.5)$$

The most likely model Θ is now determined by maximizing the log-likelihood

$$\Theta^* = \arg \max_{\Theta} \sum_{\tau \in \mathcal{T}} \log p(\tau | \Theta). \quad (3.6)$$

Optimizing this log-likelihood directly is unfeasible. Therefore, we resort to the EM algorithm (Bishop, 2006), which finds a locally optimal model Θ by iterating between computing the expectation over the latent variables and maximizing the model parameters.

In our approach, the EM algorithm repeatedly maximizes the auxiliary function

$$\begin{aligned} \Theta &= \arg \max_{\Theta} Q(\Theta, \Theta'), \\ Q(\Theta, \Theta') &= \sum_{\tau \in \mathcal{T}} \sum_{\mathcal{S} \in \mathcal{D}} p(\mathcal{S} | \tau, \Theta') \log(p(\mathcal{S}) p(\tau | \Theta, \mathcal{S})) \end{aligned} \quad (3.7)$$

until convergence. In this formulation, Θ' denotes the model parameters found in the previous iteration. The prior over the segmentation

$$\begin{aligned} p(\mathcal{S}) &= p_c \prod_{s \in \mathcal{S}} p(c_s), \\ p(c_s) &= (1 - p_c)^{c_s} p_c, \end{aligned} \quad (3.8)$$

is defined as a product of priors over transition points $p(c_s)$, where c_s is the number of possible transition points the segment s spans over. The constant $0 < p_c < 1$ defines how probable it is that a transition is a true positive transition. For $p_c < 0.5$

segments which span over multiple transition points are preferred, whereas $p_c > 0.5$ indicates that shorter segments are preferable.

Solving Equation 3.7 is computationally expensive, since the number of segmentations, $|\mathcal{S}| = 2^{|\mathcal{C}|}$, grows exponentially with the number of transitions $|\mathcal{C}|$.

However, we can reformulate Equation 3.7 such that it sums over all possible segments $s \in \mathcal{A}$, which are only quadratic in the number of transitions $|\mathcal{A}| = 0.5(|\mathcal{C}| + 1)(|\mathcal{C}| + 2)$. Inserting Equation 3.4 and Equation 3.8 into Equation 3.7, moving the log inside the product and dropping the constant term yields

$$Q(\Theta, \Theta') = \sum_{\tau \in \mathcal{T}} \sum_{\mathcal{S} \in \mathcal{D}} p(\mathcal{S} | \tau, \Theta') \sum_{s \in \mathcal{S}} \log(p(c_s) p(s | \Theta)). \quad (3.9)$$

Pulling the coefficient $p(\mathcal{S} | \tau, \Theta')$ inside the third sum and subsequently swapping the second and the third sum results in a weighted maximum-a-posteriori formulation

$$\begin{aligned} Q(\Theta, \Theta') &= \sum_{\tau \in \mathcal{T}} \sum_{s \in \mathcal{A}} \alpha_s \log(p(c_s) p(s | \Theta)), \\ \alpha_s &= \sum_{\mathcal{S} \in \mathcal{D}_s} p(\mathcal{S} | \tau, \Theta'). \end{aligned} \quad (3.10)$$

After swapping the sums we first iterate over all possible segments $s \in \mathcal{A}$ and then over the segmentations \mathcal{S} . It is important to make sure that we only iterate over segmentations that contain the respective segment $\mathcal{S} \in \mathcal{D}_s$, since it would not be an equivalent transformation of $Q(\Theta, \Theta')$ otherwise. The set of segmentations that contain s is defined as $\mathcal{D}_s = \{\mathcal{S} | s \in \mathcal{S}, \forall \mathcal{S} \in \mathcal{D}\}$.

By moving the log into the product additive constant $\sum_{\tau \in \mathcal{T}} \sum_{s \in \mathcal{A}} \alpha_s \log p(c_s)$ is obtained. The constant only depends on Θ' and not on the argument of the maximization Θ , and, hence, can be dropped yielding

$$Q(\Theta, \Theta') = \sum_{\tau \in \mathcal{T}} \sum_{s \in \mathcal{A}} \alpha_s \log p(s | \Theta). \quad (3.11)$$

The segment weighting α_s determines how likely the segment s belongs to the optimal segmentation $s \in \mathcal{S}^*$, given the current model estimate Θ' . The EM algorithm iteratively computes the weighting α_s in the E-Step, according to Equation 3.10, and updates the current model estimate, $\Theta' \leftarrow \Theta$, in the M-Step by choosing a Θ that maximizes Equation 3.11.

Expectation Step: Computing the Probability of the Segments.

In the E-Step, the segment weighting α_s , as described in Equation 3.10 is updated, and, therefore, the segments $s \in \mathcal{A}$ in Equation 3.11 are re-weighted. The weighting in Equation 3.10 is computed by summing over all segmentations $\mathcal{S} \in \mathcal{D}_s$, which contain the segment s . Computing α_s according to Equation 3.10 is therefore still of exponential complexity with respect to the number transitions $|\mathcal{C}|$. However, the segment weighting α_s can be computed much more efficiently, by reformulating Equation 3.10 slightly. Applying Bayes Theorem on Equation 3.10 yields

$$\alpha_s = \sum_{S \in \mathcal{D}_s} \frac{p(\tau | S, \Theta') p(S)}{p(\tau | \Theta')}. \quad (3.12)$$

We can pull the denominator out of the sum since it is a constant with respect to the summation, yielding

$$\begin{aligned} \alpha_s &= \frac{1}{Z} \sum_{S \in \mathcal{D}_s} p(\tau | S, \Theta') p(S) \\ Z &= p(\tau | \Theta'). \end{aligned} \quad (3.13)$$

Note, that the prior $p(S)$ as defined in Equation 3.8 does not depend on the parameters Θ' . Inserting Equation 3.4, Equation 3.5 and Equation 3.8 into Equation 3.13 results in

$$\begin{aligned} \alpha_s &= \frac{1}{Z} \sum_{S \in \mathcal{D}_s} \prod_{s' \in S} f(s'), \\ Z &= \sum_{S \in \mathcal{D}} \prod_{s' \in S} f(s'), \\ f(s') &= p(c_{s'}) p(s' | \Theta'), \end{aligned} \quad (3.14)$$

where Z denotes the normalizing constant.

Such a formulation is well studied in Graphical Models and can be solved efficiently using message passing algorithms (Bishop, 2006). In Equation 3.14 we iterate over all possible segmentations containing the segment s , $S \in \mathcal{D}_s$. By definition, \mathcal{D}_s is a combination of all possible segmentations preceding s denoted as $\mathcal{D}_s^{\rightarrow}$, s itself, and all possible segmentations succeeding s denoted as $\mathcal{D}_s^{\leftarrow}$. The sets $\mathcal{D}_s^{\rightarrow}$ and $\mathcal{D}_s^{\leftarrow}$ are illustrated as gray boxes in Figure 3.3 with $s = s_{1,2}$ for a trajectory with four possible transition points, including the start and end of the trajectory. The indices denote at which transition points the segment starts and ends. The blue segments occur alongside $s_{1,2}$ in at least one segmentation, and therefore, have to be considered when computing the segment weighting $\alpha_{s_{1,2}}$. In contrast, the red segments already include $s_{1,2}$, and therefore, must not be considered. We can rewrite Equation 3.14 as

$$\alpha_s = \frac{1}{Z} \delta_s^{\rightarrow} f(s) \delta_s^{\leftarrow}, \quad (3.15)$$

$$\delta_s^{\rightarrow} = \sum_{S \in \mathcal{D}_s^{\rightarrow}} \prod_{s' \in S} f(s'), \quad (3.16)$$

$$\delta_s^{\leftarrow} = \sum_{S \in \mathcal{D}_s^{\leftarrow}} \prod_{s' \in S} f(s'). \quad (3.17)$$

The terms δ_s^{\rightarrow} and δ_s^{\leftarrow} are defined over all possible preceding and succeeding segmentations respectively, which are still exponential in the number of transition points. We can, however, reformulate both terms as messages along a factor graph, resulting

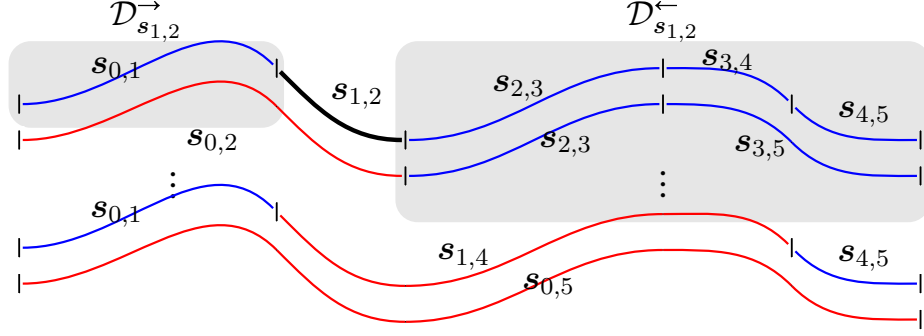


Figure 3.3.: The figure illustrates several segmentations of a one dimensional trajectory with four potential transition points, including the start and end of the trajectory. The indices of the segments denote at which transition point the segment begins and at which it ends. Assuming the segment $s_{1,2}$ is of interest, the blue segments occur alongside $s_{1,2}$ in at least one segmentation. The red segments contain $s_{1,2}$ and can therefore not occur in the same segmentation. The gray areas illustrate all possible preceding and succeeding segmentations of $s_{1,2}$, denoted as $\mathcal{D}_{s_{1,2}}^{\rightarrow}$ and $\mathcal{D}_{s_{1,2}}^{\leftarrow}$ respectively.

in a significantly lower computational complexity. Each segment represents a node in a factor graph, where each segment has a dedicated factor node and segments that share a possible transition point are additionally connected via factor nodes. Figure 3.4 shows the factor graph corresponding to the example trajectory of Figure 3.3. Analogously, $s_{1,2}$ is the segment of interest and the blue and red nodes correspond to the blue and red segments. Instead of iterating over all succeeding segmentations $\mathcal{S} \in \mathcal{D}_s^{\leftarrow}$, we can iterate over the direct successors $s' \in \mathcal{A}_s^{\text{succ}}$, and their succeeding segmentations $\mathcal{S} \in \mathcal{D}_{s'}^{\leftarrow}$. The set $\mathcal{A}_s^{\text{succ}}$ contains all segments, that begin at the transition point where s ends. Analogously, the set $\mathcal{A}_s^{\text{pred}}$ contains all segments that end immediately before s . The reformulated Equation 3.16

$$\delta_s^{\leftarrow} = \sum_{s' \in \mathcal{A}_s^{\text{succ}}} \gamma_{s'}^{\leftarrow}, \quad (3.18)$$

$$\gamma_{s'}^{\leftarrow} = f(s') \sum_{\mathcal{S} \in \mathcal{D}_{s'}^{\leftarrow}} \prod_{s'' \in \mathcal{S}} f(s''),$$

$$\gamma_{s'}^{\leftarrow} = f(s') \delta_{s'}^{\leftarrow}, \quad (3.19)$$

is now defined in terms of the backward messages, δ_s^{\leftarrow} and γ_s^{\leftarrow} of the described factor graph. The forward messages δ_s^{\rightarrow} and $\gamma_{s'}^{\rightarrow}$ are derived analogously,

$$\delta_s^{\rightarrow} = \sum_{s' \in \mathcal{A}_s^{\text{pred}}} \gamma_{s'}^{\rightarrow}, \quad (3.20)$$

$$\gamma_{s'}^{\rightarrow} = f(s') \delta_{s'}^{\rightarrow}. \quad (3.21)$$

The γ_s^\rightarrow messages are always send from segment nodes to factor nodes and contain the product of the segment probability and all incoming δ_s^\rightarrow messages. In our case each segment node has at most one incoming δ_s^\rightarrow message. The δ_s^\rightarrow messages are send from factor nodes to segment nodes and are simply the sum of all incoming γ_s^\rightarrow messages. The same applies equivalently to the backward messages γ_s^\leftarrow and δ_s^\leftarrow . Because of this representation as a sum of products this type of message passing is also referred to as sum-product algorithm. The normalizing constant Z can also be expressed in terms of either the forward or backward messages

$$Z = \sum_{s \in \mathcal{A}^{\text{start}}} \gamma_s^\leftarrow = \sum_{s \in \mathcal{A}^{\text{end}}} \gamma_s^\rightarrow,$$

where the set of segments $\mathcal{A}^{\text{start}}$ contains all possible segments that start at the beginning of the observed trajectory. Simultaneously the set \mathcal{A}^{end} contains all possible segments that end at the end of the trajectory.

The recursive nature of the messages explains the reduced computational complexity. Each forward or backward message is a combination of the incoming forward or backward messages respectively, and has to be computed exactly once. To compute the forward messages, we begin at the start nodes, $\mathcal{A}^{\text{start}}$, and pass the messages towards all of the end nodes, \mathcal{A}^{end} . To compute the backward messages we reverse the process, i.e., begin at the end nodes, \mathcal{A}^{end} , and pass the messages towards the start nodes, $\mathcal{A}^{\text{start}}$.

The segment weighting α_s can now be interpreted as the message send from the node s to the dedicated factor node, as visualized in Figure 3.5, where the left subtree consists of all preceding segments and the right subtree of all succeeding segments.

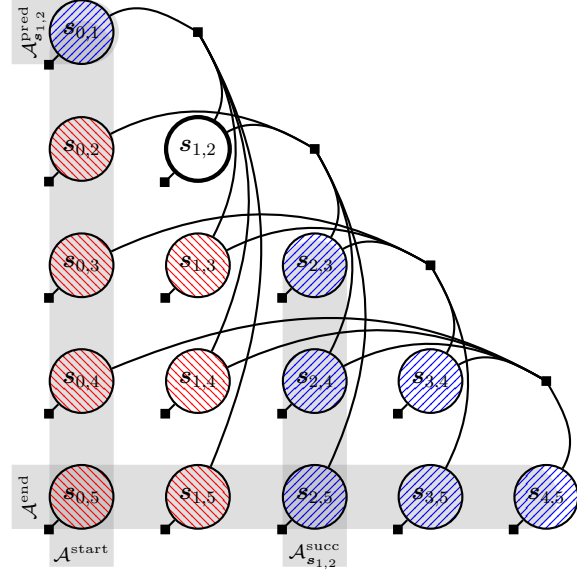


Figure 3.4.: The factor graph corresponds to an observed trajectory shown in Figure 3.3. The nodes correspond to the different segments. The segment $s_{1,2}$ is of interest and the blue and red nodes correspond to the blue and red segments. All nodes directly connected to $s_{i,j}$ are considered its neighbors. Neighbors from above are predecessors and neighbors to the right are successors, e.g., $\mathcal{A}_{s_{1,2}}^{\text{pred}} = \{s_{0,1}\}$ and $\mathcal{A}_{s_{1,2}}^{\text{succ}} = \{s_{2,3}, s_{2,4}, s_{2,5}\}$. Additionally, the sets $\mathcal{A}^{\text{start}} = \{s_{0,1}, s_{0,2}, s_{0,3}, s_{0,4}, s_{0,5}\}$ and $\mathcal{A}^{\text{end}} = \{s_{0,5}, s_{1,5}, s_{2,5}, s_{3,5}, s_{4,5}\}$ contain all segments starting at the beginning of the trajectory or ending at the end of the trajectory respectively.

Maximization Step: Learning the Movement Primitive Library.

In the maximization step, the model parameters Θ are updated by maximizing $Q(\Theta, \Theta')$. We assume that all observed demonstrations were generated by the same

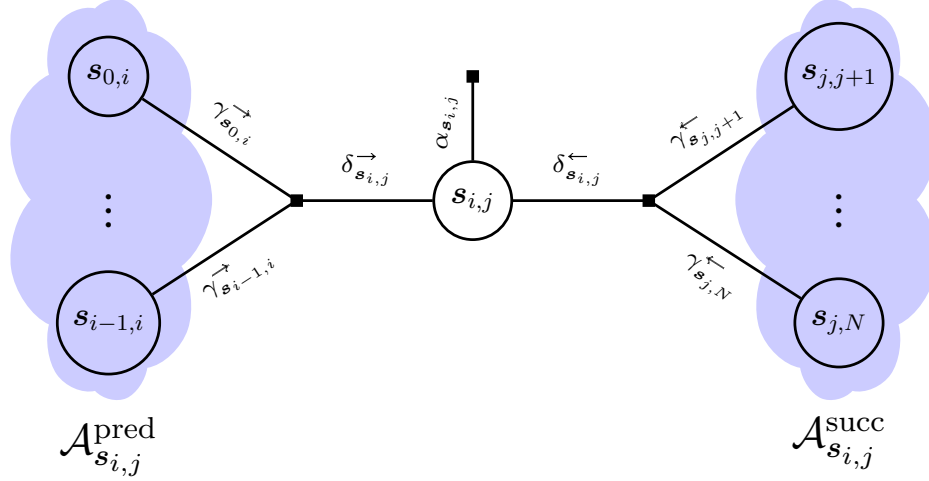


Figure 3.5.: The factor graph illustrates the computation of the segment weighting $\alpha_{s_{i,j}}$, formulated as message passing. The $\delta_{s_{i,j}}^{\rightarrow}$ and $\delta_{s_{i,j}}^{\leftarrow}$ messages are the sums of the incoming $\gamma_{s_{0..i-1,i}}^{\rightarrow}$ and $\gamma_{s_{j,j+1..N}}^{\leftarrow}$ messages. The messages $\gamma_{s_{0..i-1,i}}^{\rightarrow}$ and $\gamma_{s_{j,j+1..N}}^{\leftarrow}$ are passed by the preceding and succeeding nodes respectively, $\mathcal{A}_{s_{i,j}}^{\text{pred}}$ and $\mathcal{A}_{s_{i,j}}^{\text{succ}}$. The weighting $\alpha_{s_{i,j}}$ is the product of the incoming forward and backward messages time $f(s_{i,j})/Z$, as described in Equation 3.15.

underlying model, implying that the model update considers all possible segments independently of their corresponding demonstration.

Following (Paraschos et al., 2013), we update the model based on the projected segments rather than the segments directly. Therefore, the maximization step is defined over the set of all projected segments of all observed trajectories

$$\mathcal{W} = \{\mathbf{w} \mid \mathbf{w} = \mathbf{v}(s), \forall s \in \mathcal{A}_{\tau}, \forall \tau \in \mathcal{T}\}, \quad (3.22)$$

where \mathbf{v} is the ridge regression defined in Equation 2.3 and the set \mathcal{A}_{τ} contains all possible segments of the observed trajectory τ . Besides the significantly lower dimensionality of each projected segment \mathbf{w} compared to s , working in the projected space has the advantage of comparing different segments time invariantly, purely based on their shape. Therefore, similar segments, which mainly differ in their execution speed, will be assigned to the same primitive. Accordingly, we define α_s over the projected segments

$$\alpha_{\mathbf{w}} = \alpha_s \iff \mathbf{w} = \mathbf{v}(s).$$

Given these definitions, the reformulated auxiliary function

$$Q_{\mathcal{W}}(\Theta, \Theta') = \sum_{\mathbf{w} \in \mathcal{W}} \alpha_{\mathbf{w}} \log p(\mathbf{w} \mid \Theta), \quad (3.23)$$

$$p(\mathbf{w} \mid \Theta) = \sum_{k=1}^{|\mathcal{M}|} \lambda_k p(\mathbf{w} \mid \theta_k), \quad (3.24)$$

takes the form of a weighted log-likelihood, where $p(\mathbf{w} \mid \Theta)$ is defined as standard Gaussian Mixture Model (GMM). Because \mathcal{W} was defined as a set which contains

all the projected segments for all trajectories $\tau \in \mathcal{T}$, the sum over \mathcal{W} replaces both sums over τ and s .

It is unknown which projected segment belongs to which movement primitive, resulting in $p(\mathbf{w} | \Theta)$ being a mixture model with latent variables. Therefore, it is unfeasible to maximize $Q(\Theta, \Theta')$ directly with respect to the parameters Θ .

It is, however, possible to estimate Θ by maximizing the weighted maximum log-likelihood. Given our model we can again apply a weighted EM algorithm for GMMs. Instead of maximizing the log-likelihood for a single projected segment $\log p(\mathbf{w} | \Theta)$, we can maximize the auxiliary function

$$Q_{\mathbf{w}}(\Theta, \Theta') = \sum_{k=1}^{|\mathcal{M}|} \beta_{k,\mathbf{w}} \log \lambda_k p(\mathbf{w} | \theta_k),$$

where the mixing coefficients $\sum_{k=1}^{|\mathcal{M}|} \lambda_k = 1$ sum up to one and $\beta_{k,\mathbf{w}} = p(k | \mathbf{w}, \Theta')$ are typically referred to as responsibilities. Replacing $\log p(\mathbf{w} | \Theta)$ for $Q_{\mathbf{w}}(\Theta, \Theta')$ in Equation 3.23,

$$Q_{\mathcal{W}}(\Theta, \Theta') = \sum_{\mathbf{w} \in \mathcal{W}} \alpha_{\mathbf{w}} \sum_{k=1}^{|\mathcal{M}|} \beta_{k,\mathbf{w}} \log \lambda_k p(\mathbf{w} | \theta_k),$$

and rearranging the sums yields the standard auxiliary function of a weighted EM for GMMs

$$Q_{\mathcal{W}}(\Theta, \Theta') = \sum_{k=1}^{|\mathcal{M}|} \sum_{\mathbf{w} \in \mathcal{W}} \alpha_{\mathbf{w}} \beta_{k,\mathbf{w}} \log \lambda_k p(\mathbf{w} | \theta_k). \quad (3.25)$$

Since the set \mathcal{W} is defined across all observed trajectories, the model will contain primitives learned from segments of different observed trajectories. The EM algorithm for GMMs is widely applied and well known. In the expectation step, the responsibilities are updated

$$\beta_{k,\mathbf{w}} = \frac{\lambda'_k p(\mathbf{w} | \theta'_k)}{\sum_{k'=1}^{|\mathcal{M}|} \lambda'_{k'} p(\mathbf{w} | \theta'_{k'})},$$

which emerges from applying Bayes theorem.

Maximizing the auxiliary function $Q_{\mathcal{W}}(\Theta, \Theta')$ represents a constrained optimization problem, which can be solved by applying the method of Lagrangian multipliers. Solving the Lagrangian leads to the model updates performed in the maximization step. The updates for each parameter are given by

$$\begin{aligned} \lambda_k &= \frac{\sum_{\mathbf{w} \in \mathcal{W}} \alpha_{\mathbf{w}} \beta_{k,\mathbf{w}}}{\sum_{\mathbf{w} \in \mathcal{W}} \alpha_{\mathbf{w}}}, \\ \boldsymbol{\mu}_k &= \frac{\sum_{\mathbf{w} \in \mathcal{W}} \alpha_{\mathbf{w}} \beta_{k,\mathbf{w}} \mathbf{w}}{\sum_{\mathbf{w} \in \mathcal{W}} \alpha_{\mathbf{w}} \beta_{k,\mathbf{w}}} \text{ and} \\ \boldsymbol{\Sigma}_k &= \frac{\sum_{\mathbf{w} \in \mathcal{W}} \alpha_{\mathbf{w}} \beta_{k,\mathbf{w}} (\mathbf{w} - \boldsymbol{\mu}_k)(\mathbf{w} - \boldsymbol{\mu}_k)^T}{\sum_{\mathbf{w} \in \mathcal{W}} \alpha_{\mathbf{w}} \beta_{k,\mathbf{w}}}. \end{aligned}$$

Algorithm 1: Probabilistic Segmentation

input : The initial set of transition points \mathcal{C}

output : The underlying mixture model Θ^*

The underlying segmentation \mathcal{S}^*

K : current number of clusters

L : current labeling of the segments

while not converged **do**

E-Step : compute the weighting α_s as described in Equation 3.15

M-Step: 1. compute \mathcal{W} according to Equation 3.22

 2. determine K and L by applying Gaussian-means on \mathcal{W}

 3. update Θ^* by using a weighted EM-GMM on \mathcal{W} with K
 clusters, initial labeling L and weights α_w

A known disadvantage of EM for mixture models is that the number of components is generally assumed to be known a priori. Additionally, the number of movement primitives $|\mathcal{M}|$ in our approach depends on the latent segmentation, and can therefore change every time we change the weighting of the segments. We circumvent this problem by using the Gaussian-means algorithm (Hamerly and Elkan, 2003) to determine the number of movement primitives, including an initial labeling for the EM algorithm. The Gaussian-means algorithm is a bisecting k -means algorithm which uses a test based on the Anderson-Darling statistic to determine if the data assigned to a cluster is Gaussian or not. If the data is not Gaussian, the cluster is split. We summarize our Probabilistic Segmentation (ProbS) method in Algorithm 1.

Convergence of ProbS and Complexity of the Auxiliary Function $Q(\Theta, \Theta')$.

The maximization step of the proposed method is itself an EM for GMMs, which is known to converge to a local maximum of the log-likelihood. Furthermore, maximizing $Q_{\mathcal{W}}(\Theta, \Theta')$ will at least find a local maximum for $Q(\Theta, \Theta')$, which guarantees the convergence of the proposed method (Wu, 1983). Due to the number of segmentations $|\mathcal{D}| = 2^{|\mathcal{C}|}$, the initial formulation in Equation 3.7 is in $\mathcal{O}(2^{|\mathcal{C}|})$. After the reformulations given in Equation 3.11 and Equation 3.15 the problem is defined over the segments. The number of segments is quadratic in the number of transitions, $|\mathcal{A}| = 0.5(|\mathcal{C}| + 1)(|\mathcal{C}| + 2)$, and therefore, computing Equation 3.11 is in $\mathcal{O}(|\mathcal{C}|^2)$. The reformulation therefore allows to consider significantly more transitions.

3.4 Experimental Evaluation of ProbS

We evaluated our method on a real robot writing task. We compared it to a baseline method, Expectation-Maximization algorithm Gaussian mixture models, and a state-of-the-art non-parametric segmentation method, Beta Process Autoregressive Hidden Markov Model, (Fox, 2009), (Niekm et al., 2012). In (Lioutikov et al., 2015) we per-



(a) While holding a pen, the robot was demonstrated writing trajectories on a white board using kinesthetic teaching.



(b) The robot executes a trajectory which was sequenced from the learned movement primitive library.

Figure 3.6.: The experimental setup of the real robot writing task. The executed sequence is not restricted to three letter words, except by the whiteboard. The observed trajectories were demonstrated by kinesthetic teaching. Subsequently a movement primitive library was learned by segmenting the trajectories using ProbS. Finally sequences of the learned movement primitive library were executed on the real robot platform.

formed a real robot chair assembly task. In order to show that useful movement primitives were extracted, we replayed the movement primitives on the real robot, while conditioning the movement primitives to assemble the chair in and undemonstrated order. In both tasks the experimental platform was a seven DoF KUKA lightweight arm equipped with a five finger DLR HIT Hand II as end effector, as shown in Figure 3.1. Additionally, we segmented multiple demonstrations of a barrett wam robot platform returning balls in table tennis setup. In this task ProbS successfully identified primitives for forehand and backhand swings.

3.4.1 Setup of the Real Robot Writing Task

In the robot writing task the robot was demonstrated how to write trajectories as continuous motions via kinesthetic teaching, while holding a pen and drawing on a white board, as shown in Figure 3.6a. A total of 27 words were demonstrated. Each word based on a three letter alphabet, containing the letters, “a”, “u” and “y”. The recorded data were the two dimensional trajectories on the whiteboard-plane. The initial over-segmentation was produced by a curvature-based heuristic, where the transition points were identified by the curvature exceeding a certain threshold. The heuristic resulted in 11 transitions per observation leading to a total of 55296 possible segmentations or equivalently 2106 segments. ProbS reduced the number of active transition points significantly, as illustrated in Figure 3.7. In average ProbS reduced the number of transition points to four per observation.

In order to demonstrate the advantage of optimizing both the segmentation and the movement primitive library iteratively, we chose an Expectation-Maximization algorithm over Gaussian mixture models (EM-GMM) as a baseline, where the number

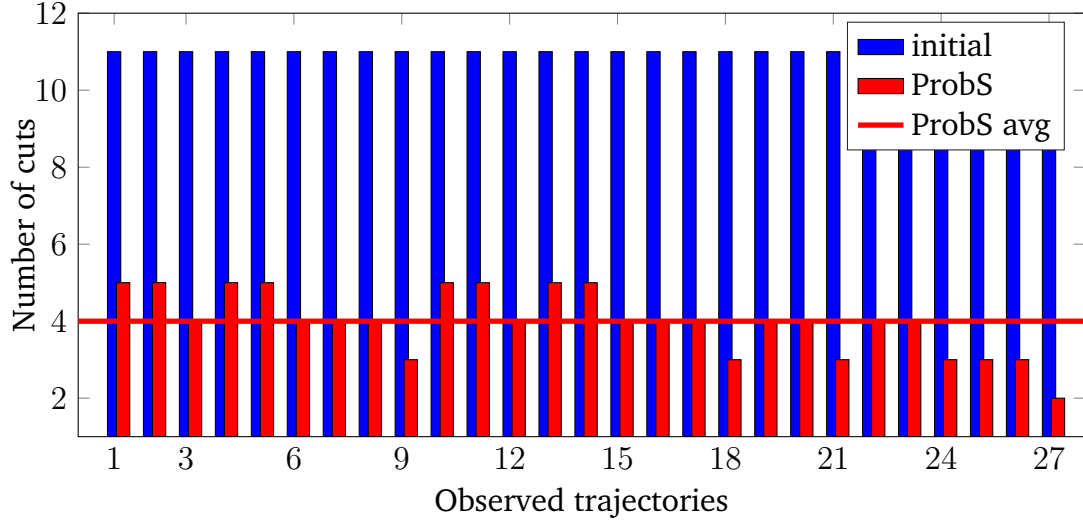


Figure 3.7.: ProbS is able to reduce the number of active transitions significantly, leading to a compact representation of the observations.

of clusters as well as the initial labeling is determined by the Gaussian-means algorithm (Hamerly and Elkan, 2003). In addition, we compared our method to the state-of-the-art, non-parametric segmentation method Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM), as applied in (Fox, 2009) and (Niekum et al., 2012). Given the found number of movement primitives and the initial labeling of the Gaussian-means algorithm, the EM-GMM baseline identified a total of eight movement primitives. The BP-AR-HMM method does not rely on an initial segmentation, however, due to the sampling process, the algorithm is computationally very expensive. In our evaluation the best solutions were found with an autoregressive order of two. Furthermore, the segmentation had to be performed on the velocity of the trajectories, since the method was otherwise not able to identify the same movement primitive at different absolute positions. The final segmentation identified seven movement primitives.

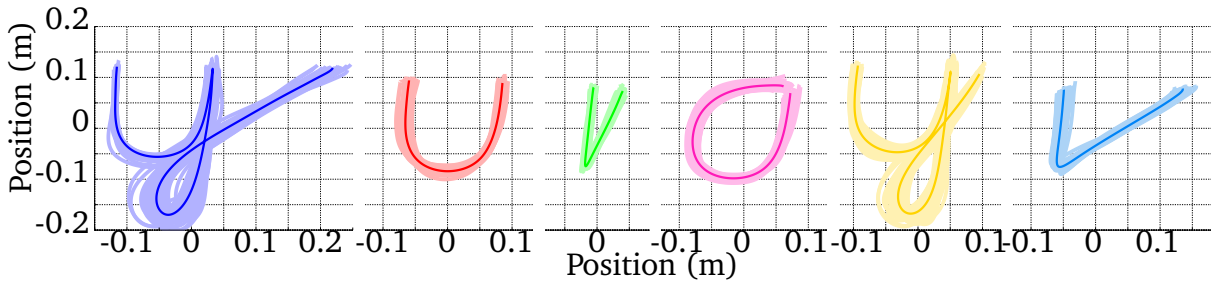


Figure 3.8.: The six movement primitives learned by ProbS. From left to right. 1: The letter “y” when proceeding an “a”. 2: The corpus of an “u”. 3: The tail of the letters “a” and “u” when not followed by an “a”. 4: The corpus of an “a”. 5: the letter “y” when not proceeding an “a”. 6: The tail of the letters “a” and “u” when followed by an “a”.

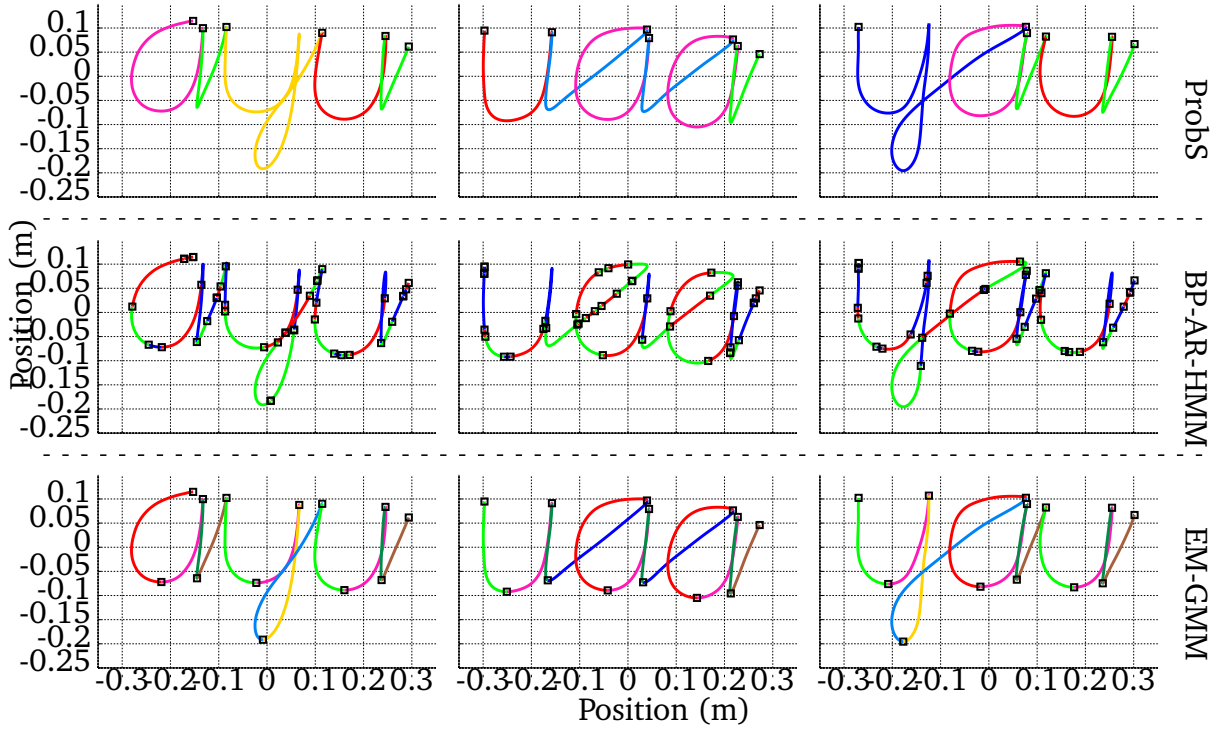


Figure 3.9.: Comparison of ProbS, BP-AR-HMM, EM-GMM on the letter segmentation task. All methods resulted in a similar number of movement primitives (8, 6, 7), but the resulting segmentations differed significantly. Same color within one row indicates the assignment to the same movement primitive.

ProbS was able to identify six underlying movement primitives, which are shown in Figure 3.8. The brighter colored background illustrates the variance of each primitive. In each demonstration, the letter “a” begins further right than “u” or “y” and is therefore preceded by a longer tail than the other letters. ProbS separated those variations for “a” and “u” into their corpus and the two different tails, leading to the movement primitives 2,3,4 and 6. Movement primitives 1 and 5 show the two variations of the letter “y”. These variations were not merged into a single primitive, because the merged skill would achieve a significantly lower average likelihood across all “y”-segments than the two learned primitives achieve for their respective “y”-segments. Additional demonstrations in between those two variants would most likely yield a single “y” primitive.

Comparison on the Letter Segmentation.

A comparison of the methods is shown in Figure 3.9. While ProbS and the EM-GMM benefit from the initial segmentation, the BP-AR-HMM found very different segments. However, many of the segments found by BP-AR-HMM occur in very different shapes and lengths, which results in movement primitives with high variance. Given the curvature based heuristic the segments found by ProbS and the EM-GMM appear semantically meaningful and have a high recall value throughout the observations.

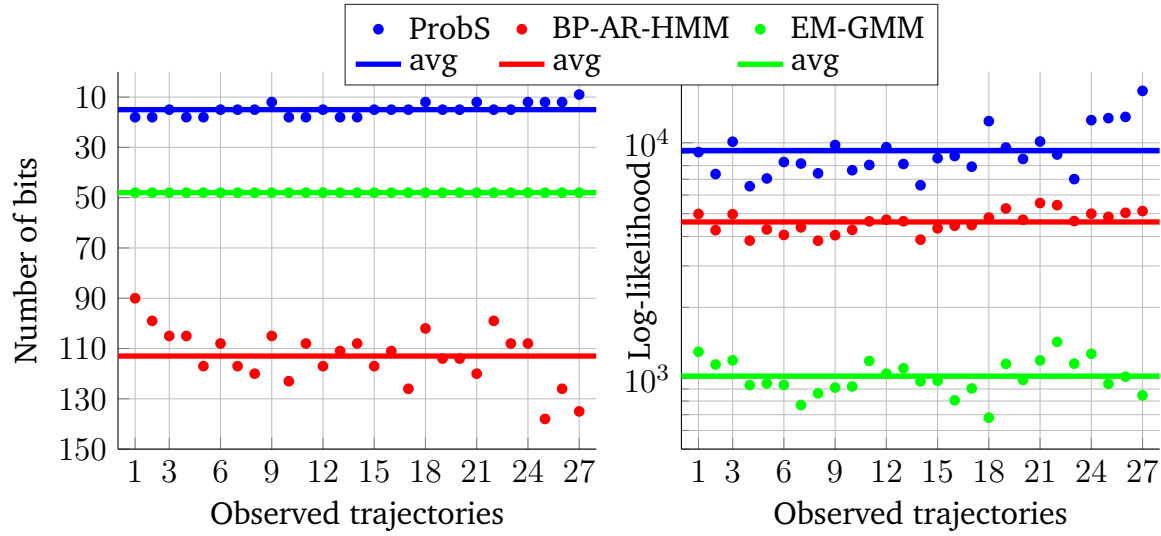


Figure 3.10.: (left) The quality of the found segments is quantified by the number of bits each observations requires to be encoded. Less is better. (right) The learned movement primitives are evaluated using a LOOCV. ProbS shows the advantage of including the movement primitive learning in the segmentation process.

Encoding Lengths of the Different Segmentations.

As a metric to quantify the value of the segmentation we computed the number of bits necessary to encode each observed trajectory τ given the identified segments $N_\tau = |\mathcal{S}_\tau|N_{\mathcal{M}}$. \mathcal{S}_τ is the learned segmentation for trajectory τ and $N_{\mathcal{M}} = \lceil \log_2 |\mathcal{M}| \rceil$ denotes the number bits necessary to encode each learned primitive uniquely. The coding length for each observation as well as the average, is shown in Figure 3.10. As the EM-GMM does not change the number of segments in any observation the encoding solely depends on the predefined number of cluster, which in this case leads to a fixed length of 48 bits per observation. By reducing the number of active transitions and learning a small set of movement primitives ProbS achieved an average encoding length of 15 bits. While BP-AR-HMM found a similar number of movement primitives, each observation was explained by significantly more segments than in either the EM-GMM or ProbS. BP-AR-HMM achieved a high average encoding length of 113 bits. This evaluation shows, that ProbS was able to find segments, which allow a compact representation of the observations.

Leave-One-Out Cross Validation of the Learned Movement Primitives.

Another important aspect of ProbS is the quality of the learned movement primitives. We compared to the other two methods by applying a leave-one-out cross validation (LOOCV) on each of the learned segmentations. Thus, learning the movement primitives excluding one observation and subsequently computing the average log-likelihood of each segment in the excluded observation given the learned movement primitives. Let \mathcal{S}_τ and $\mathcal{D}_\tau = \{\mathcal{S}_\tau | \forall \tau \in \mathcal{T}\}$ be the learned segmentation for trajectory τ and respectively the set of all learned segmentations for the set of

trajectories \mathcal{T} . The set $\mathcal{D}_{\mathcal{T} \setminus \tau} = \mathcal{D}_{\mathcal{T}} \setminus \mathcal{S}_{\tau}$ then denotes the set of all learned segmentations $\mathcal{D}_{\mathcal{T}}$ except \mathcal{S}_{τ} . The average log-likelihood for the LOOCV is now given as $\bar{\ell} = 1/|\mathcal{T}| \sum_{\tau \in \mathcal{T}} \sum_{s \in \mathcal{S}_{\tau}} \log p(s | \Theta_{\mathcal{D}_{\mathcal{T} \setminus \tau}})$, where $\Theta_{\mathcal{D}_{\mathcal{T} \setminus \tau}}$ denotes the mixture of primitives learned from $\mathcal{D}_{\mathcal{T} \setminus \tau}$.

The results of the LOOCV are shown in Figure 3.10. The BP-AR-HMM outperformed the EM-GMM significantly, showing its superiority with an average of 4610 over the simple clustering method with an average of 1021. However, given its intrinsic optimization of the learned movement primitives, ProbS achieved a significantly higher average log-likelihood of 9272. At the same time, ProbS had the advantage of using additional knowledge given by the initial heuristic. In future work ProbS could replace the heuristic by a non-parametric method, similar to BP-AR-HMM, and hence, improve the so proposed segmentation.

Writing Using the Learned Movement Primitive Library.

Finally, we show the applicability of our method in real robot scenarios, by executing a trajectory composed of the learned movement primitives. We are not restricted to a particular trajectory, but can randomly sample from our set of movement primitives, shown in Figure 3.8. Furthermore, we are not limited to reproducing the observations but can sequence an arbitrary number of movement primitives from the learned library. Additionally, we can take advantage of the various properties of MPs, e.g., choosing a different duration for each movement primitive and conditioning on certain positions. Figure 3.6 shows the execution of a sampled sequence. The sequence was randomly sampled from the mixture model. A transition table was learned from the segmented demonstrations to ensure a feasible sequence of selected components.

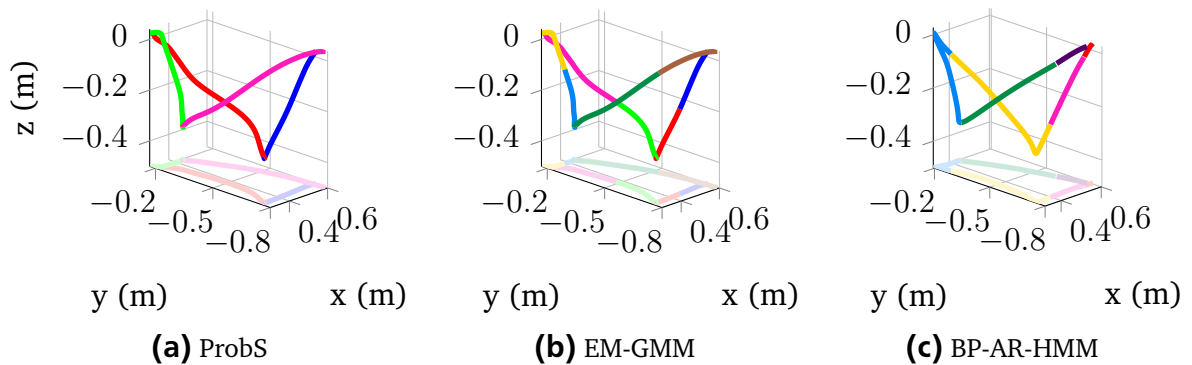


Figure 3.11.: The first demonstrations explained by the three different methods. Different colors within each plot illustrated different movement primitives. ProbS explains the demonstration with four movement primitives, EM-GMM identified eight movement primitives in the demonstration and BP-AR-HMM separated the demonstration into a total of 9 segments. Some of the segments have a very short duration and are not visible in the plot.

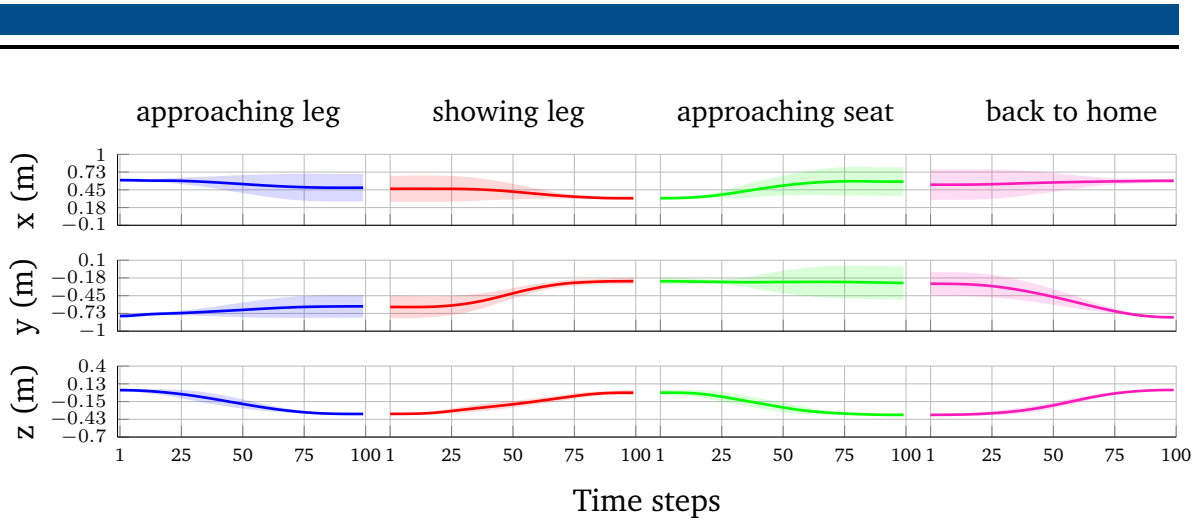


Figure 3.12.: The first three dimensions of the four movement primitives learned by ProbS to exemplify the skills, omitting the remaining dimensions. The dark line shows the mean and the shaded area corresponds to two times the standard deviation. The movement primitives are semantically meaningful, i.e., approaching a leg, showing the leg to a camera, approaching the seat and going back to the home position.

The initial over-segmentation was produced by a velocity based heuristic where the transition points were positioned at the extrema of the velocity profile for each observation.

The heuristic resulted in nine transitions per observation leading to a total of 768 possible segmentations or equivalently 216 segments. This experiment shows the feasibility of the learned movement primitives and the applicability of ProbS in real robot tasks.

Comparison on the Chair Assembly.

We also compared ProbS to EM-GMM and BP-AR-HMM on the chair assembly task. Figure 3.11 shows the identified segments of each method for the first demonstration.

ProbS was able to identify four underlying movement primitives. Each of the movement primitives occurs exactly once in every demonstration and corresponds to one of the four steps of each demonstration. The first three dimensions are shown in Figure 3.12. It is clearly visible that the movement primitives preserve a characteristic shape while the variance at certain time steps shows the adaptability of the movement primitive at those points.

Given the found number of movement primitives and the initial labeling of the Gaussian-means algorithm, the EM-GMM baseline identified a total of eight movement primitives. Analogously to ProbS each found movement primitive is present exactly once in each demonstration.

In this task BP-AR-HMM achieved the best results with an autoregressive order of one and identified four primitives. The method also explained similar segments in different demonstrations by the same primitives. However, it also identified very short segments within the demonstrations and assigned it to the same primitive as other much longer and differently shaped segments.

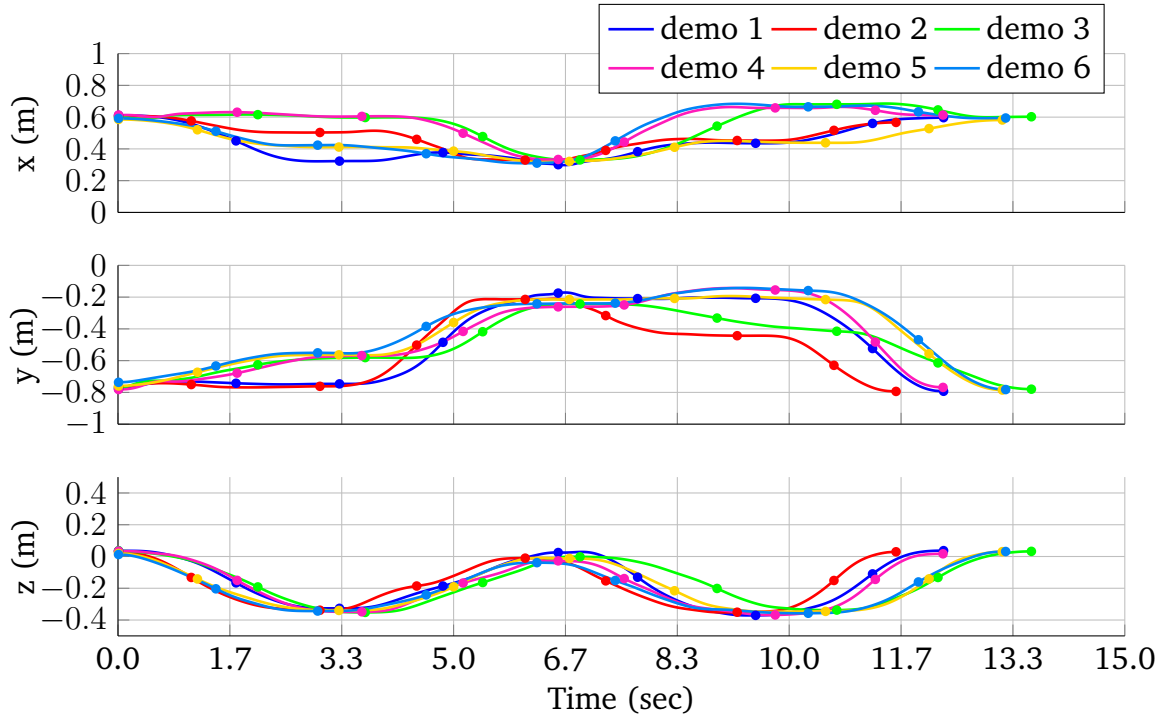


Figure 3.13.: The first three dimensions, corresponding to the Cartesian position, of the six observations. Each observation demonstrates the insertion of a chair leg into a hole in the seat. The transitions determined by the initial heuristic are illustrated as dots.

3.4.2 Setup of the Real Robot Chair Assembly Task

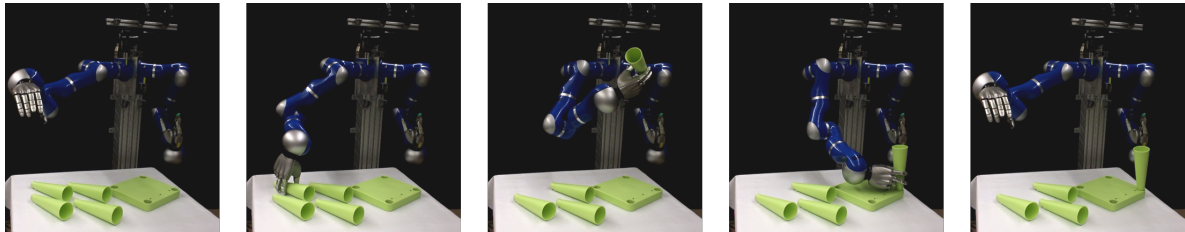
The chair assembly was demonstrated by a human, as seen in Figure 3.13. The wrist of the human was tracked using the OptiTrack motion capture system. A total of six demonstrations were performed, where each demonstration consisted of four phases, i.e., approaching and picking up a chair leg, showing the leg tip to the camera, approaching and inserting the leg into the seat and finally returning to the home position. Each data point is a seven dimensional vector in Cartesian space containing the three dimensional wrist position and the four dimensional orientation encoded as a quaternion. The tracked positions are shown in Figure 3.13.

Chair Assembly using the Learned Movement Primitives.

We show the applicability of our method in real robot scenarios, by assembling an Ikea chair using the learned movement primitive library. As shown in Figure 3.14, the robot is able to extract the necessary movement primitives from the given demonstrations. Similarly, to the writing experiment each segment was drawn from the mixture model, while a learned transition table ensured that a sensible sequence of primitives was queried. The start and end point of each movement primitive were conditioned to the corresponding point of interest, e.g., the "inserting" movement primitive was conditioned to the hole position. Since each movement primitive was learned from only six samples, the variance at some points was too low to successfully condition



(a) Demonstration of the chair assembly recorded via the OptiTrack motion capture system.



(b) Execution of the learned movement primitives.

Figure 3.14.: Demonstration and execution of the chair assembly task. The human was tracked during the chair assembly. The observed demonstrations were subsequently segmented using ProbS. The movement primitives of the learned library were subsequently sequenced and executed on a robot platform to assemble the chair.

the corresponding ProMP. Therefore, we scaled the covariance matrix of each movement primitive artificially. This scaling was only necessary for the execution and did not occur during the learning of the library. This experiment shows that ProbS is able to segment entire demonstrations to extract meaningful movement primitives. These movement primitives can be used and sequenced in order to solve observed as well

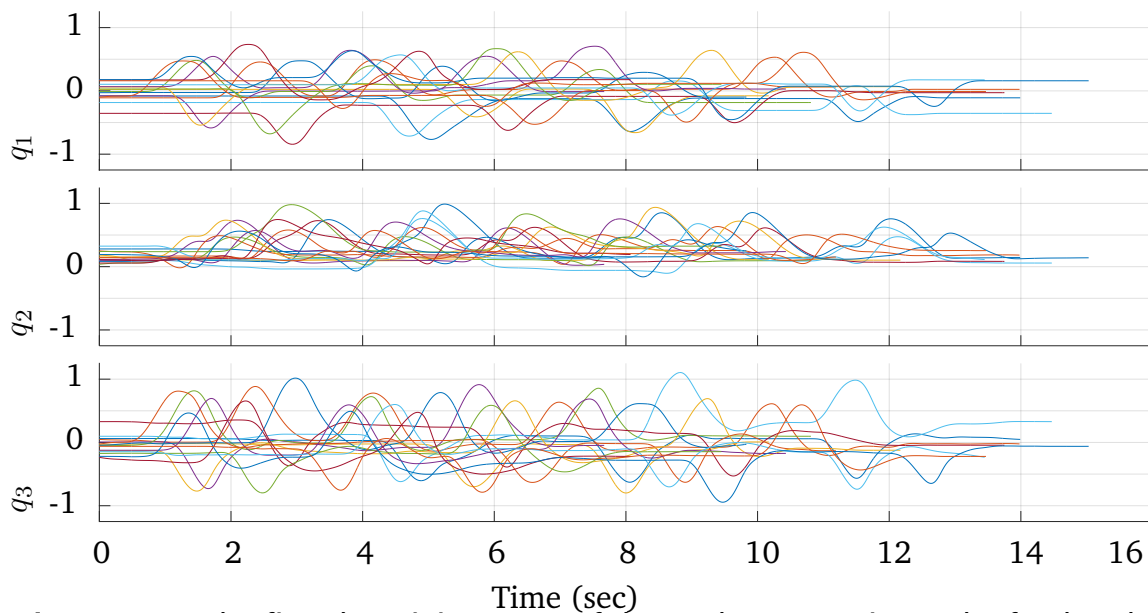


Figure 3.15.: The first three joint states of the 16 demonstrations. The forehand and backhand swings occur randomly and are unequal in duration. Additionally, the duration between swings is not fixed and can be considered a waiting period.

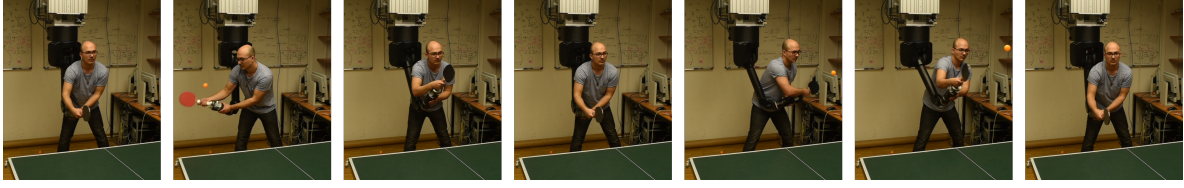


Figure 3.16.: Kinesthetic teaching of the barrett wam robot platform for the table tennis task. While one person throws table tennis balls towards the robot, another person moves the robot manually in order to return the balls. During the teaching all seven joints are recorded for the subsequent segmentation.

as new tasks. For example, the chair was assembled with combinations of legs and holes which were not present in the demonstrations.

3.4.3 Robot Table Tennis

In this task multiple demonstrations of table tennis forehand and backhand swings were segmented into a total of four movement primitives. The swings were demonstrated on a 7 DoF barrett wam robot with a racket as an end effector. While one person threw table tennis balls in the direction of the robot another person applied kinesthetic teaching to return the balls with the robot, as shown in Figure 3.16. During the demonstrations the joint states were continuously recorded with a sampling rate of 100 Hz. A total of 20 forehand and 26 backhand swings were recorded randomly across 16 demonstrations in overall 189 seconds. The demonstrations for the first three joints are shown in Figure 3.15. The initial over-segmentation was given by a low velocity heuristic which resulted in a transition whenever all joints were below a certain threshold. A total of 127 transitions were initially detected with an average of 7.9 transitions per demonstration. ProbS learned a total of four primitives, a forehand swing, a backhand swing and two waiting primitives. For both waiting primitives the mean for each joint changes at most 0.03 radians, which effectively results in an almost steady racket. However, the two primitives

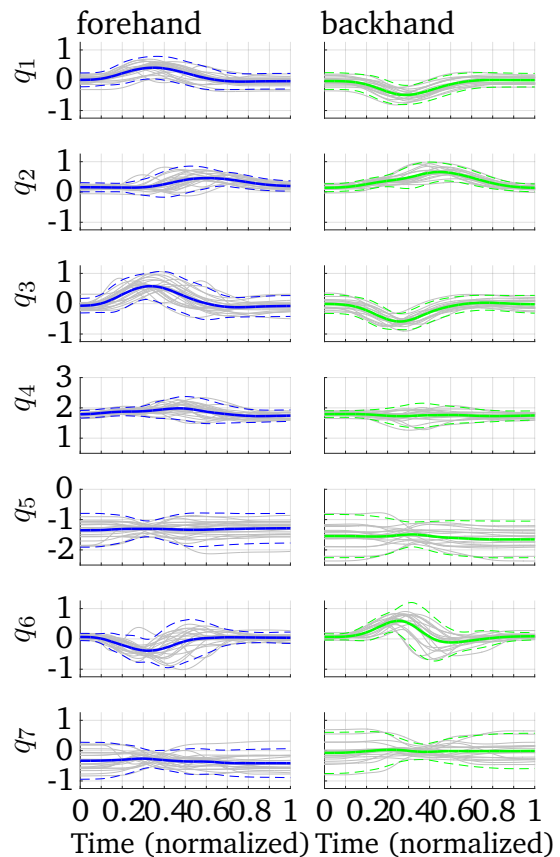


Figure 3.17.: Mean and 2x std for each joint. The gray lines are the segments used to learn the primitives. For illustration purposes the is was normalized. This normalization is not used in the ProbS algorithm.

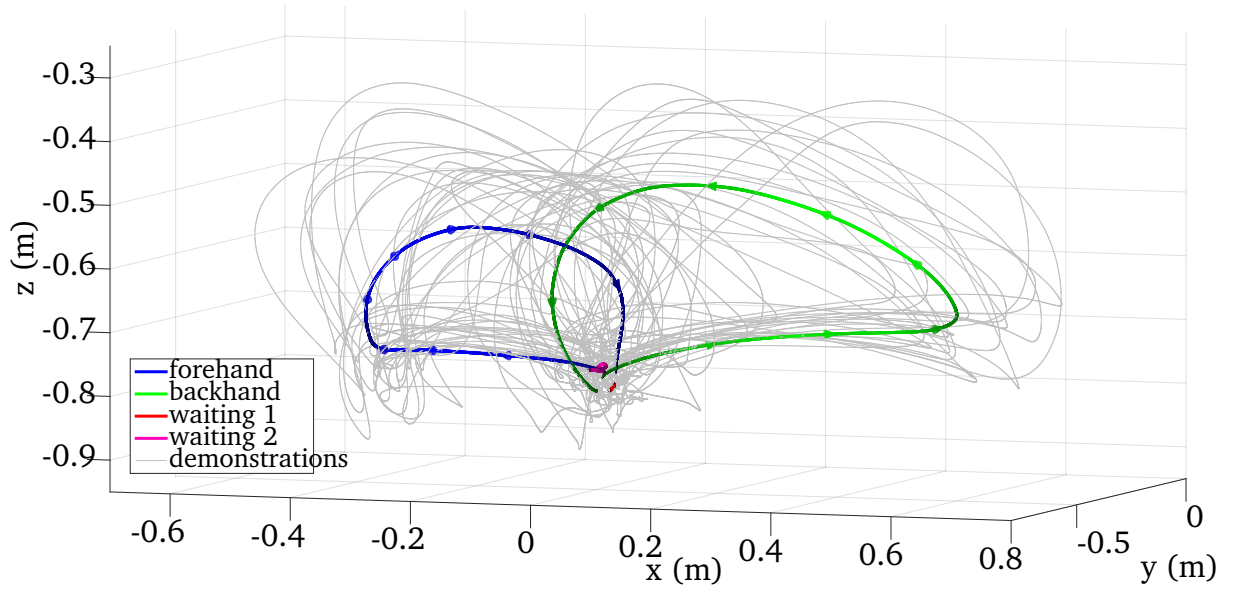


Figure 3.18.: The four learned primitives for the robot table tennis task. The trajectories are shown in Cartesian space and were computed by applying the forward kinematics to the mean of each corresponding primitive. The brightness represents the velocity of the trajectory and the arrows indicate the direction of the movement. The grey lines show the original demonstration.

contain the most movement in different joints, and hence, were not learned as a single primitive. The forehand and backhand primitives show characteristic joint movements, as seen in Figure 3.17. To illustrate the learned primitives the Cartesian trajectories of the racket were computed by applying forward kinematics to the mean joint trajectories of the four primitives. The resulting movements are shown in Figure 3.18 alongside the original demonstrations.

3.5 Conclusion

In this chapter we proposed a new algorithm for the segmentation of unlabeled trajectories. The algorithm builds a movement primitive library that is used to infer correct segmentations. The library as well as the inferred segmentations of the trajectories are iteratively optimized as there is a high dependency between both entities. We presented an efficient formulation of the algorithm, transforming multiple steps from exponential to quadratic complexity. Our algorithm takes advantage of heuristics that are used to over-segment the trajectories. In comparison to other state-of-the-art methods such as non-parametric auto-regressive HMMs, our algorithm has less hyper-parameters to fine tune and clear computational advantages. Furthermore, the returned movement primitive library looked much more compact than the ones retrieved with related approaches. The evaluation of the algorithm showed that the method can be applied in real world scenarios in both Cartesian and joint space, such as the presented chair assembly task and the robot table tennis task.

In this work we did not attempt to learn libraries across various tasks. Given that both the E-Step and the M-Step treat the segments independent of the underlying task, the performance of ProbS should not change significantly. However, learning primitive libraries across multiple tasks will be part of future work. Given that the presented method clusters similar segments into the same primitive, it is assumed that segments that belong to the same primitive result in small intra-cluster distances. In the case of very few observed segments, however, the method might learn separate primitives for segments that appear to belong together, since there are not sufficient samples between the found sub-clusters, and, hence, the applied G-means algorithm decides for multiple clusters instead of a single one. A goal of future work will be to investigate alternative approaches to determine the number of primitives. In addition, we will concentrate on reducing the reliance on the heuristics and investigate alternative discriminative and generative approaches for change point detection. Another potential improvement of interest is the substitution of the inner EM loop through variational Bayesian inference. The in order to of the inner loop through a variational i as well as learning high level control variables of each movement primitive, such as possible conditioning points.

4 Learning Attribute Grammars from Movement Primitives Sequences

In the previous chapter we presented an approach to segment demonstrations of an entire task into a set of movement primitives. Each of the learned primitives represents an atomic, simple movement and is, therefore, appropriate for tasks consisting of a single stroke-based or rhythmic movement (Paraschos et al., 2017a). Single movement primitives (MPs) have been used in a large variety of applications, e.g., table tennis (Muelling et al., 2013), pancake flipping (Kormushev et al., 2010) and hockey (Paraschos et al., 2017a). However, for more complex tasks a single MP is often not sufficient. Such tasks require sequences of MPs for feasible solutions. Considering a set or library of MPs, such sequences can be generated in a variety of ways, including Hidden Markov Models (Kulic et al., 2012), Mixture Models (Lioutikov et al., 2017a) and other hierarchical approaches (Stulp and Schaal, 2011). These approaches can be regarded as mechanisms that produce sequences of MPs, revealing a common, important downside: understanding these mechanisms requires a significant amount of expert knowledge. However, a declared goal of robotics is the deployment of robots into scenarios where direct or indirect interactions with non-expert users are required. Therefore, more intuitive sequencing mechanisms for non-experts are necessary.

This chapter introduces the use of formal grammars for the sequencing of MPs. In particular, we focus on probabilistic context-free grammars (PCFGs) and propose a method to induce PCFGs from observed sequences of primitives. Formal grammars represent a formal description of symbols and rules, encoding the structure of a corresponding language. They have been extensively studied in both natural language processing and compiler construction but have also been applied in a variety of fields such as molecular biology (Chiang et al., 2006), bioinformatics (Rivas and Eddy, 2000), computer vision (Zhu and Mumford, 2007; Kitani et al., 2005) and robotics (Dantam and Stilman, 2013; Lee et al., 2013; Sarabia et al., 2015). The choice of learning a probabilistic representation over a single deterministic plan is based on the insight that probabilistic representations of behavior are generally more robust to changes than deterministic representations, especially in dynamic environments. For instance, in a collaborative task a fixed plan describing the behavior between a robot and a user would require the user to always behave according to the set plan. A distribution over plans allows for at least some flexibility as long as the plan is still within the distribution. Furthermore, PCFGs allow the implicit embedding of hierarchies within the rules of the grammar associating every produced sequence with at least one corresponding parse tree. Such a parse tree represents the derivation of the produced sequence in an intuitive way. Figure 5.1 shows a learned grammar for placing a stone in a game of tic-tac-toe, including the parse tree for a produced primitive sequence.

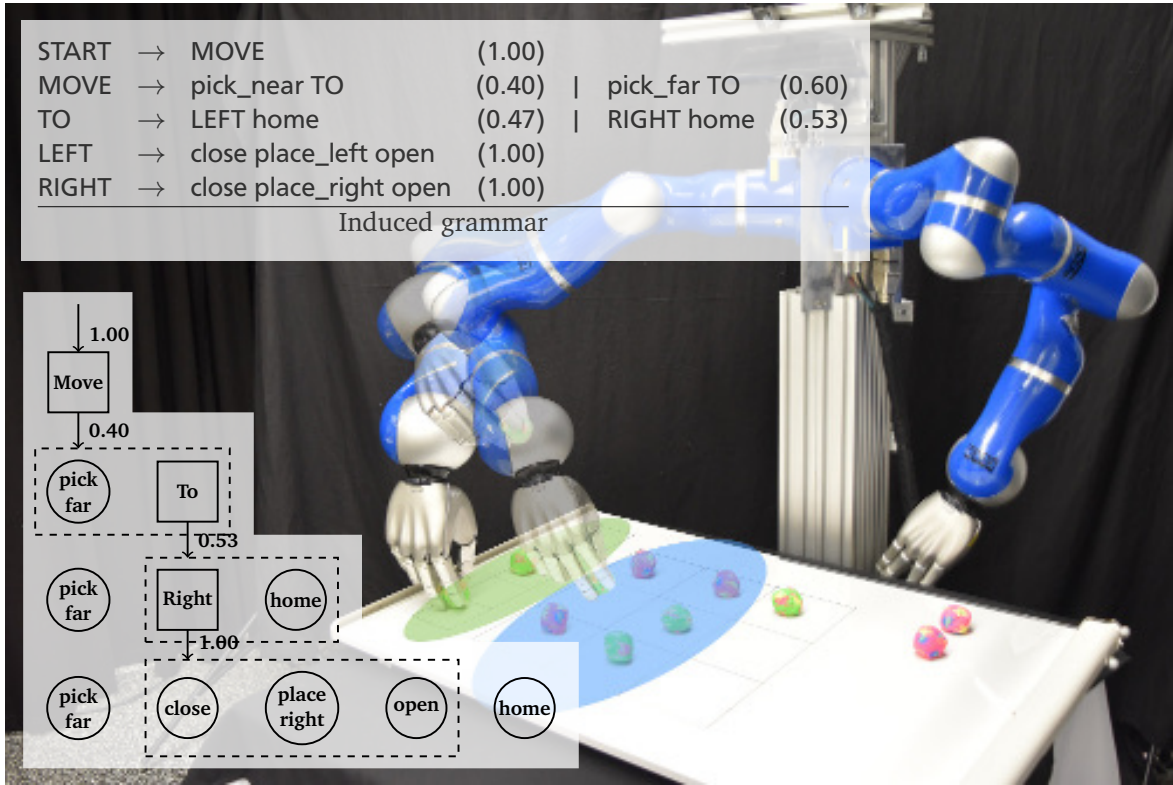


Figure 4.1.: The robot executes a turn in the tic-tac-toe game, represented as a sequence of movement primitives. The sequence was generated by a probabilistic context-free grammar learned from previously labeled observations.

The understandability of the grammar itself highly depends on the size and structure of the grammar. The induction of concise but expressive grammars is considered non-trivial and in the context of natural language even an ill-posed problem. A common approach to grammar induction is to formulate the problem as a search problem where each possible grammar is a node in the search space and a set of operators generate the edges between those nodes. This search space can then be traversed through different search methods where a scoring function determines the quality of each grammar. Stolcke et al. (Stolcke, 1994) suggested formulating the problem as a Maximum-a-posteriori estimation where the scoring is defined as the posterior given the observations. In order to reduce the possibility of getting stuck in bad local optima, the search space was traversed via beam search. In this work we formulate the search as a Markov chain Monte Carlo (MCMC) optimization similarly to (Talton et al., 2012), where the scores are defined as posteriors over the grammars given the observations.

To ease the learning of grammars the proposed approach exploits the structure inherently presented in the physical motions. We assume each segment of the observed sequences to be a sample from an underlying library of movement primitives (e.g., (Lioutikov et al., 2017a; Niekum et al., 2015)). Due to the considerably smaller size of a primitive library compared to the corpus of a natural language, the observed sequences of even complex tasks show a simpler structure than a sentence of a natural language. Furthermore, the category of a movement primitive, e.g., hand or arm

movement, can be easier deduced than the category of a word, e.g., verb or noun. We discuss how to determine the category of a movement primitive later in this chapter.

An important restriction that improves the induction of grammars for movements is that any produced sequence has to result in a continuous trajectory inside the state space. Therefore, any grammar that would produce a jump in the state space is invalid and has to be removed from consideration. In this work we avoid such grammars directly by restricting the operators to exclusively produce valid grammars, by ensuring the connectivity between two consecutive movement primitives.

The contributions of this work are the induction of probabilistic context-free grammars for the sequencing of movement primitives. The posteriors are computed using a novel prior distribution that avoids many disadvantages of existing methods based on minimum description length and Dirichlet distributions. The search is formulated as a Markov chain Monte Carlo optimization where the proposed distributions are defined through restrictions put upon the operators connecting the grammar search space. These restrictions include physical constraints presented in the domain of movements.

4.1 Related Work

Movement primitives are usually used to solve tasks consisting of single, atomic stroke-based or periodic movements (Paraschos et al., 2017a). For more complex tasks, however, a sequence of primitives has to be applied. An example of such a task is the grasping, positioning, and cutting of a vegetable (Lioutikov et al., 2014) with Dynamical Movement Primitives (DMPs) (Ijspeert et al., 2013a). However, in (Lioutikov et al., 2014) the sequences were not learned, but predefined. An approach combining the segmentation of observations and the learning of a sequencing mechanism is presented in (Kulic et al., 2012). The primitives are encoded using Hidden Markov Models and a graph structure is learned during the segmentation. This graph can be used subsequently to sequence the primitives. Another approach featuring a sequence graph was presented in (Manschitz et al., 2014). The graph is learned from demonstrations through an agglomerative clustering scheme. In this work, we propose probabilistic context-free grammars as a means of sequencing movement primitives. Grammars bring the advantage of being a general method capable of representing hierarchies in a principled and intuitive manner.

Motion grammars (Dantam and Stilman, 2013) are extensions of context-free grammars modeling the discrete and continuous dynamics of hybrid systems (Dantam and Stilman, 2012). Motion grammars aim at fast task verification and have not yet been induced from observations. In (Dantam et al., 2011) and (Sarabia et al., 2015) the benefits of applying context-free grammars in human-robot interaction scenarios such as playing chess or making music collaboratively. In (Lee et al., 2012; Lee et al., 2013) probabilistic context-free grammars are used to sequence discrete actions. Analogously to (Stolcke, 1994) the grammar is learned by applying a beam search for the maximal posterior inside the grammar space. The grammar space is traversed by applying the merge and chunk operators (Stolcke, 1994) to observed sequences. In contrast to (Stolcke, 1994) a n-gram like frequency table is used to determine reoccurring patterns in the observations, hence, identifying candidate productions for the

chunk operator. To avoid unintuitive, compact grammars the prior definition, originally defined solely by the minimal description length, was extended by a log-Poisson term similar to (Kitani et al., 2008). The meaning of the operators shall become clear later in the chapter.

While sharing the motivation of learning intuitive, probabilistic context-free grammars for primitive sequencing, our work differs from (Lee et al., 2012) in several ways. We use a stochastic movement primitive representation and actively take advantage of its properties to induce the grammar. We deviate from the common structure prior definition as an exponential distribution over the minimal description length and define the entire prior as a combination of several Poisson distributions. Furthermore, we use Markov chain Monte Carlo optimization to find the grammar maximizing the posterior, similarly to (Talton et al., 2012), which is more robust to local optima than beam-search.

The grammar induction approach described in (Talton et al., 2012) uses the Metropolis-Hastings algorithm (Andrieu et al., 2003) to learn grammars describing designs in various domains, such as websites and geometric models. The prior is defined using the description length and the grammar learning is not used in any robotics context. In addition, the structure of the observed sequences differs significantly from our problem setting. In (Talton et al., 2012), the observations and, hence, the starting points of the grammar induction are already hierarchical structures. Therefore, it is sufficient to traverse the grammar space using solely the merge and split operators. These operators allow the generalization and specialization of grammars, but are not able to introduce new hierarchies like the chunk operator (Stolcke, 1994). In this work, we apply all three operators. To achieve the required irreducibility of the Markov chain we additionally introduce the insert operator, negating the effects of the chunk operator.

4.2 Problem Statement

Given a set of demonstrations $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_{|\mathcal{D}|}\}$ a set of primitives $\Theta = \{\theta_1, \theta_2, \dots, \theta_{|\Theta|}\}$, each demonstration represents a labeled sequence of primitives $\mathbf{d}_i \in \Theta^+$. The goal of this work is to learn an Attribute grammar $\mathcal{G}_{\text{att}}^*$ that is concise and expressive yet has an easily comprehensible structure. For instance, given a set of demonstrations of turns in a game of tic-tac-toe

$$\mathcal{D} = \left\{ \begin{array}{l} (\text{pick_far, close, place_right, open, home}), \\ (\text{pick_near, close, place_right, open, home}), \\ \vdots \\ (\text{pick_far, close, place_left, open, home}), \\ (\text{pick_near, close, place_left, open, home}) \end{array} \right\},$$

a concise representation of the possible sequences is given by the grammar

START	→	MOVE	(1.00)		
		MOVE.stone = START.stone			
		MOVE.field = START.field			
MOVE	→	pick_near TO	(0.40)		pick_far TO (0.60)
		pick_near.stone = MOVE.stone			pick_far.stone = MOVE.stone
		TO.field = MOVE.field			TO.field = MOVE.field
TO	→	LEFT home	(0.47)		RIGHT home (0.53)
		LEFT.field = TO.field			RIGHT.field = TO.field
LEFT	→	close place_left open	(1.00)		
		place_left.field = LEFT.field			
RIGHT	→	close place_right open	(1.00)		
		place_right.field = RIGHT.field.			

(GR 4.1)

Grammar 4.1 represents a generalized structure over the observed demonstrations and allows the sampling of new sequences while the attributes allow for the adaptation of individual primitives. For instance, the attribute `stone` contains the position of the stone which is supposed to be played next and the primitives, e.g., `pick_near` and `pick_far` can now be conditioned on the passed down position. In addition, attributes are introduced that ensure a smooth and continuous trajectory across each sampled primitive sequence despite the adaptation of individual primitives.

The desired grammar is learned by inducing a PCFG \mathcal{G}^* from the demonstrations \mathcal{D} and enhancing it afterwards with a general attribute scheme for sequencing movement primitives $\mathcal{G}^* \xrightarrow{\text{att}} \mathcal{G}_{\text{att}}^*$. The set of terminals is defined as the set of primitives $\mathcal{A} = \Theta$ and during the learning of the grammar the terminals, and, hence the primitives are considered immutable, implying that the search space consists of grammars that only differ in \mathcal{S} , \mathcal{V} or \mathcal{R} . Each grammar represents a node in the grammar space \mathfrak{G} , as illustrated in Figure 4.2, where the directed edges between nodes are defined by operators. Operators manipulate the rule set \mathcal{R} of a grammar \mathcal{G} and consequently create a new grammar \mathcal{G}' while the grammar space itself is explored via a Markov chain Monte Carlo optimization. In order to optimize for grammars with concise and comprehensible structures a novel prior based on Poisson distributions is introduced. The grammar induction and the prior are presented in later in the chapter.

Given that the sequences produced by the grammar directly result in the movement of a robot it is important that there are no state jumps at the transition between two consecutive primitives. Throughout this thesis this requirement is referred to as primitive connectivity. The connectivity of two primitives depends on the category of each primitive and the transition overlap between the primitives. The category of a primitive classifies which degrees of freedom are effectively controlled by the primitive. Primitives assigned to disjoint categories are not subject to the connectivity requirement. The transition overlap describes how much the end of one primitive and the beginning of the next primitive overlap. If the overlap is too small a smooth transition is unlikely and the connectivity requirement is violated. Next, we present a method for identifying the categories of primitive. Afterwards we introduce an approach to compute the transition overlap between two subsequent ProMPs.

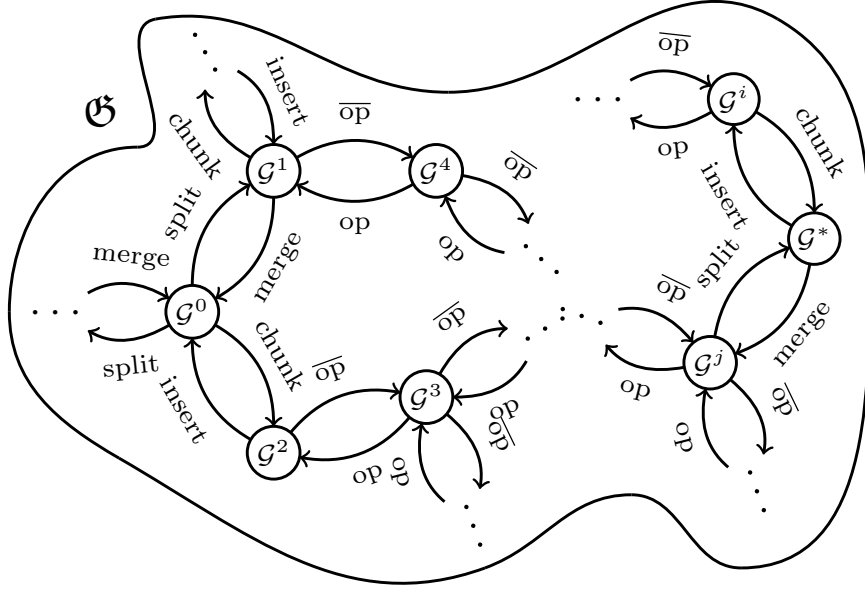


Figure 4.2.: The grammar space \mathcal{G} contains all valid grammars $\mathcal{G}^0 \dots \mathcal{G}^*$. The space is traversed by applying operators $\text{op} \in \mathcal{O} = \{\text{merge}, \text{split}, \text{chunk}, \text{insert}\}$ on the current grammar. For every operator op generating \mathcal{G}' from \mathcal{G} , there exists an $\overline{\text{op}}$ that generates \mathcal{G} from \mathcal{G}' , e.g., $\overline{\text{merge}} = \text{split}$, $\overline{\text{chunk}} = \text{insert}$.

4.3 Identifying the Primitive Category

Given a robotic platform with independent kinematic chains, e.g., an arm and a hand, each of these chains represents a category of movements. Such categories allow the relaxation of the connectivity requirement to hold only between primitives of the same category. Hence, the connectivity requirement of two subsequent primitives of the same category is independent of any primitive executed between them that is does not belong to that category. Given that some primitives might contain significant movement across categories, we treat the identification of the primitive category as a simple multi-label classification problem. The classification is based on the degrees of freedom that are active during the movement and assigns each primitive θ to a category \mathcal{C} .

In this work each primitive is represented as a single ProMP. The distribution over the trajectory τ given the parameters θ is defined in Equation 2.5. Assuming equidistant time steps, the distribution

$$p(\dot{\tau}) = \prod_t \mathcal{N}(\dot{\tau} \mid \dot{\Phi}_t \mu_w, \dot{\Phi}_t \Sigma_w \dot{\Phi}_t^T + \Sigma_{\text{obs}}), \quad (4.1)$$

describes the velocity trajectories of the corresponding ProMP, with $\dot{\Phi}_t$ being the first time derivative of the basis functions and Σ_{obs} denoting the observation noise with respect to the velocity trajectory. In order to identify the active degrees of freedom we analyze the mean and standard deviation of the velocity distribution. In particular,

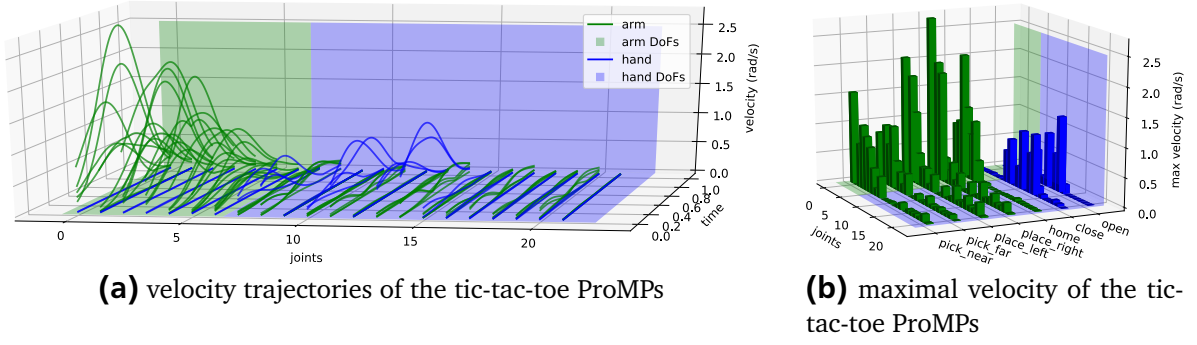


Figure 4.3.: (a) shows the $|\text{mean}| + 2 \text{ std}$ of the velocity distribution defined in Equation 4.1 for each ProMP, while (b) shows the maximal velocities as computed in Equation 4.2. The colors indicate which categories were assigned to each ProMP and background colors highlight to which category each joint belongs.

we are interested in the maximal absolute velocity over the time steps and define the maximum velocity feature as

$$\psi^{\text{vel}}(\theta) = \max_t \left(|\dot{\Phi}_t \mu_w| + 2\sqrt{\text{diag}(\dot{\Phi}_t \Sigma_w \dot{\Phi}_t^T + \Sigma_{\text{obs}})} \right). \quad (4.2)$$

A primitive is considered active with respect to the category \mathcal{C}_i if and only if any of the corresponding elements in $\psi^{\text{vel}}(\theta)$ is above the threshold ϵ_i , i.e.,

$$\theta \in \mathcal{C}_i \iff \bigvee_{d \in \text{dof}(\mathcal{C}_i)} \psi_d^{\text{vel}}(\theta) > \epsilon_i, \quad (4.3)$$

with $\text{dof}(\mathcal{C}_i)$ being the set of degrees of freedom associated with category \mathcal{C}_i and $\psi_d^{\text{vel}}(\theta)$ is the maximal velocity of the d^{th} degree of freedom. The threshold ϵ_i is chosen manually.

In the tic-tac-toe task, two different categories are distinguished, arm movements and hand movements. If two subsequent hand movements are connectible it does not matter how many arm movements are sequenced in between them. The connectibility requirement is still fulfilled. At the same time hand movements can now be executed at arm configurations at which the hand movement has not been observed in the demonstrations. Figure 4.3 shows the velocity trajectories of the seven primitives used in the tic-tac-toe task where the colors of the trajectories indicate the identified category and the background color highlights the degrees of freedom associated with each category. Given the demonstrations, every primitive was assigned exactly one category, even though the described approach allows the association of multiple categories to a single primitive. While the given example could also have been solved by a simple annotation of the observed data, examples can be thought of where the presented automated annotation is of great benefit. For instance, a significantly larger set of observed sequences or tasks that require multiple interacting kinematic chains, e.g., a bi-manual task.

4.4 Determining Connectivity of Primitives

In this section we discuss how to automatically determine if two ProMPs are connectible, that is if two subsequent ProMPs would result in a jump in the state space or not. Given that ProMPs are a probabilistic trajectory representation it seems fitting to use probabilistic similarity measures, such as the Kullback-Leibler divergence or the Hellinger distance to test if two ProMPs are connectible. However, considering that avoiding jumps in the state space is a spatial requirement probabilistic similarity measures can be misleading. We solve the connectibility problem in the spatial domain by treating each ProMP as a multidimensional tube. At every time step t a ProMP θ_i can be approximated by a hyper ellipsoid

$$\mathcal{E}_t(\theta_i) = (\mathbf{c}_t = \Phi_t \boldsymbol{\mu}_w, \mathbf{S}_t = \Phi_t \boldsymbol{\Sigma}_w \Phi_t^T + \boldsymbol{\Sigma}_{\text{obs}})$$

with center \mathbf{c}_t and shape matrix \mathbf{S}_t . Two ProMPs, θ_i and θ_j , are now considered connectible if the transition overlap between their corresponding tubes is larger than a predefined threshold

$$\theta_i \xrightarrow{\text{con}} \theta_j \iff \text{overlap}(\theta_i, \theta_j) \geq \epsilon_{\text{overlap}}.$$

The transition overlap from θ_i to θ_j is defined over the last ellipsoid of the preceding ProMP $\mathcal{E}_T(\theta_i)$ and the first ellipsoid of the succeeding ProMP $\mathcal{E}_1(\theta_{i+1})$

$$\text{overlap}(\theta_i, \theta_j) = \frac{\text{vol}(\mathcal{E}_T(\theta_i) \cap \mathcal{E}_1(\theta_j))}{\text{vol}(\mathcal{E}_T(\theta_i))},$$

with vol denoting the volume and $\mathcal{E}_T(\theta_i)$ and $\mathcal{E}_1(\theta_j)$ being the last ellipsoid of θ_i and the first primitive of θ_j respectively. Hence, the transition overlap describes the percentage of the end of θ_i that is covered at the beginning of θ_j .

Unfortunately computing the volume of the intersection between two hyper ellipsoids is considered #P-complete (Bringmann and Friedrich, 2010). We circumvent this problem by computing the overlap for each degree of freedom independently and choosing the minimal value as an approximation of the ellipsoidal overlap. As discussed in the previous section the connectibility between two ProMPs is only considered if both ProMPs share at least one primitive category \mathcal{C}

$$\text{overlap}(\theta_i, \theta_j) \approx \min_{d \in \text{dof}(\mathcal{C})} \frac{|\text{intersec}(\theta_i, \theta_j, d)|}{|c_{i,T,d} \pm n s_{i,T,d}|},$$

$$\text{intersec}(\theta_i, \theta_j, d) = c_{i,T,d} \pm n s_{i,T,d} \cap c_{j,1,d} \pm n s_{j,1,d},$$

where $c_{i,T,d}$ and $s_{i,T,d}$ are the d^{th} element of the ellipsoid center and the d^{th} standard deviation for primitive θ_i at time step T . The constant n decides how many standard deviations wide the considered interval will be.

The threshold $\epsilon_{\text{overlap}}$ can be defined either manually or derived from observations. Given that the learned grammar should at least be capable to reproduce the initially

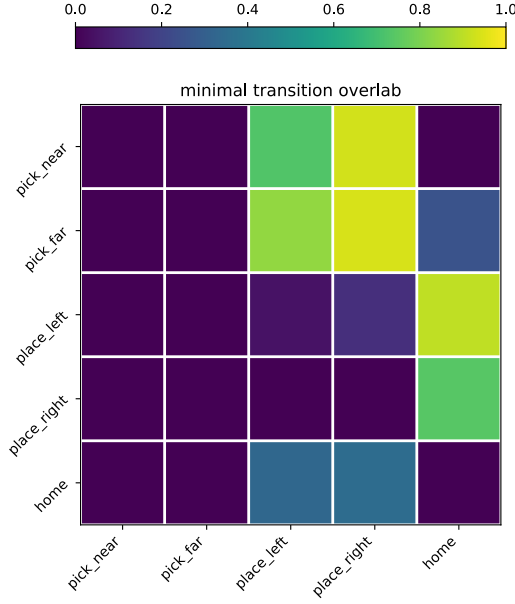


Figure 4.4: The transition overlap for each arm primitives of the tic-tac-toe task. A value above the threshold $\epsilon_{\text{overlap}} = 0.69$ signifies that the primitive at that row is connectible to the primitive of that column. The ellipsoids were $n = 2$ standard deviations wide and the threshold was determined from the observations with $\alpha = 0.95$.

given observations all ProMPs that were pairwise connected in the observations have to be considered connectible. Therefore, the threshold

$$\epsilon_{\text{overlap}} = \alpha \min_{(\theta_i, \theta_j) \in \text{pairs}(\mathcal{D})} \text{overlap}(\theta_i, \theta_j),$$

is defined as a percentage α of the minimal overlap value of all ProMPs that were connected in the observations. $\text{pairs}(\mathcal{D})$ is a function that returns all consecutive primitive pairs in the set of demonstrations \mathcal{D} . We can now define two sets for each primitive. One set contains all primitives it is connectible to $\text{Con}(\theta)$ and the other set contains all primitives it is connectible from $\text{Ccon}(\theta)$

$$\begin{aligned} \text{Con}(\theta_i) &= \left\{ \theta_j \mid \theta_j \in \Theta \wedge \theta_i \xrightarrow{\text{con}} \theta_j \right\}, \\ \text{Ccon}(\theta_i) &= \left\{ \theta_j \mid \theta_j \in \Theta \wedge \theta_j \xrightarrow{\text{con}} \theta_i \right\}. \end{aligned}$$

Figure 4.4 illustrates the transition overlap between each pair of primitives of the tic-tac-toe task arranged in an adjacency like matrix, where each row and column indicate which primitives belong into $\text{Con}(\theta)$ and $\text{Ccon}(\theta)$ respectively.

Deciding the connectibility of two primitives using their overlap rather than purely basing it on the observations has the significant advantage of allowing to connect two primitives which might not have been observed connected during the demonstrations but are nevertheless safely connectible.

4.5 Inducing PCFGs for Movement Primitives

In this section we introduce an grammar induction approach where the grammar search is defined as a maximum-a-posteriori problem and the grammar space is traversed using a Markov chain Monte Carlo approach. We introduce a novel prior over the grammar structure based on three Poisson distributions allowing to define a desired grammar structure in more detail than common grammar priors. Furthermore,

we discuss problems of common grammar priors and the advantages of the presented prior. We present four operators that allow the traversal of the grammar space and define distributions over each given a grammar. The proposal distribution of the MCMC approach is defined as a mixture over the operator distributions.

4.5.1 Learning Grammars through Posterior Optimization

The posterior $p(\mathcal{G}|\mathcal{D})$ describes how probable a given grammar \mathcal{G} is given the observed sequences \mathcal{D} . By applying Bayes theorem we can reformulate the posterior, and, hence the maximization as

$$\mathcal{G}^* = \arg \max_{\mathcal{G}} p(\mathcal{G}|\mathcal{D}) = \arg \max_{\mathcal{G}} p(\mathcal{D}|\mathcal{G}) p(\mathcal{G}), \quad (4.4)$$

where $p(\mathcal{D}|\mathcal{G})$ is the likelihood of the labeled demonstrations \mathcal{D} given the grammar \mathcal{G} . The likelihood will be presented in the next section. Afterwards we discuss common choices for the prior $p(\mathcal{G})$ and finally we introduce a novel grammar prior based on Poisson distributions.

The likelihood $p(\mathcal{D}|\mathcal{G})$

is computed for each demonstration independently, yielding

$$p(\mathcal{D}|\mathcal{G}) = \prod_{d \in \mathcal{D}} p(d|\mathcal{G}). \quad (4.5)$$

Depending on the grammar \mathcal{G} the sequence d could have been produced in multiple ways. Considering every possible derivation results in the sum-product formulation

$$p(d|\mathcal{G}) = \sum_{T \in T(d, \mathcal{G})} \prod_{(A, r, \rho) \in T} \rho,$$

where T represents a single parse tree and $T(d, \mathcal{G})$ denotes a function producing all feasible parse trees. The 3-tuple (A, r, ρ) represents an edge in the parse tree T connecting the nonterminal A and its production $r \in R_A$ with a probability of $\rho \in \rho_A$. In this work, the function T creating all possible parse trees for a given demonstration d , is implemented by the Earley parser (Earley, 1983). While the Earley parser suffers from a higher complexity compared to other parsers, it has the advantage that the parsed grammars do not have to be in any particular form.

The grammar prior $p(\mathcal{G})$

is commonly modeled as a joint distribution over the grammar probabilities $\rho_{\mathcal{G}} = \{\rho_A | A \in \mathcal{V}\}$ and the grammar structure $\mathcal{G}_{\mathcal{R}} = \{(A, R_A) | A \in \mathcal{V}\}$ (Stolcke, 1994; Kitani et al., 2008; Talton et al., 2012; Lee et al., 2013),

$$p(\mathcal{G}) = p(\rho_{\mathcal{G}} | \mathcal{G}_{\mathcal{R}}) p(\mathcal{G}_{\mathcal{R}}). \quad (4.6)$$

The conditional $p(\boldsymbol{\rho}_{\mathcal{G}} | \mathcal{G}_{\mathcal{R}})$ itself can be modeled as an independent joint distribution over the parameters of each nonterminal $A \in \mathcal{V}$,

$$p(\boldsymbol{\rho}_{\mathcal{G}} | \mathcal{G}_{\mathcal{R}}) = \prod_{\boldsymbol{\rho}_A \in \boldsymbol{\rho}_{\mathcal{G}}} p(\boldsymbol{\rho}_A). \quad (4.7)$$

The dependency on the grammar structure is implicit, since the probabilities $\boldsymbol{\rho}_A \in \boldsymbol{\rho}_{\mathcal{G}}$ depend on both the set of nonterminals \mathcal{V} and the productions for each nonterminal R_A . The parameters for each nonterminal $\boldsymbol{\rho}_A \in \boldsymbol{\rho}_{\mathcal{G}}$ form a multinomial distribution, i.e., $\sum_{\rho \in \boldsymbol{\rho}_A} \rho = 1$. Therefore, a Dirichlet distribution would be an obvious choice for the probability distribution over the parameters $p(\boldsymbol{\rho}_A)$ for a single nonterminal $A \in \mathcal{V}$. A significant drawback of using a Dirichlet distribution is its factorial growth in the dimensionality of the multinomial. In fact, using an uninformative Dirichlet distribution, i.e., setting the concentration parameters to 1.0, will result in a probability density of $p(\boldsymbol{\rho}_A) = (\dim(\boldsymbol{\rho}_A) - 1)!$ for any $\boldsymbol{\rho}_A \in \boldsymbol{\rho}_{\mathcal{G}}$.

To compensate for this growth, the structure prior $p(\mathcal{G}_{\mathcal{R}})$ is usually modeled as an exponential distribution over the minimal description length (MDL) of the grammar structure $\mathcal{G}_{\mathcal{R}}$. Every symbol in the production rules, terminal and nonterminal, contributes to the MDL with $\log_2(|\mathcal{A}| + |\mathcal{V}|)$ bits, yielding the over all description length

$$\begin{aligned} \text{MDL}(\mathcal{G}_{\mathcal{R}}) &= \sum_{(A, R_A) \in \mathcal{G}_{\mathcal{R}}} \sum_{r \in R_A} \text{MDL}(r), \\ \text{MDL}(r) &= (1 + |r|) \log_2(|\mathcal{A}| + |\mathcal{V}|). \end{aligned}$$

A prior $p(\mathcal{G}_{\mathcal{R}})$ defined as an exponential distribution over the MDL(\mathcal{G}) will prefer small and concise grammars. However, such a prior can lead to grammars that are too compact to be intuitive for non-experts. In order to prefer grammars with a desired production length, η_r , the MDL has been extended with the log of a Poisson distribution with mean η_r (Kitani et al., 2008; Lee et al., 2013). Because of the factorial growth of the parameter prior $p(\boldsymbol{\rho}_{\mathcal{G}} | \mathcal{G}_{\mathcal{R}})$ the structure prior is often additionally amplified with an exponential weighting term (Talton et al., 2012) to remain of significance for the overall grammar prior $p(\mathcal{G})$ and, hence, the posterior $p(\mathcal{G} | \mathcal{D})$.

The likelihood $p(\mathcal{D} | \mathcal{G})$ is defined as a product over the average of probabilities, which always results in $p(\mathcal{D} | \mathcal{G}) \leq 1.0$. However, the described grammar prior $p(\mathcal{G})$ is the product of two probability densities, which will very quickly result in $p(\mathcal{G}) \gg 1.0$ and therefore dominate the posterior.

The novel prior

presented in this chapter aims at inducing PCFGs which are easily understandable for non experts. The key to achieving this goal is the grammar structure, rather than the grammar parameters. Therefore, we suggest a grammar prior, that does not explicitly model a Dirichlet distribution over the parameters but instead implicitly considers the

parameters in the overall grammar prior $p(\mathcal{G})$. We model the parameter prior and the structure prior jointly $p(\mathcal{G}) = p(\boldsymbol{\rho}_{\mathcal{G}}, \mathcal{G}_{\mathcal{R}})$ as

$$p(\boldsymbol{\rho}_{\mathcal{G}}, \mathcal{G}_{\mathcal{R}}) = \frac{p(|\mathcal{R}||\eta_{\mathcal{R}})}{|\mathcal{R}|} \sum_{(A, R_A, \boldsymbol{\rho}_A) \in \mathcal{R}} \gamma(A, R_A, \boldsymbol{\rho}_A) \quad (4.8)$$

$$\gamma(A, R_A, \boldsymbol{\rho}_A) = p(|R_A||\eta_R) p(R_A|\boldsymbol{\rho}_A, \eta_r), \quad (4.9)$$

where the probabilities over the number of rules $p(|\mathcal{R}||\eta_{\mathcal{R}})$ and the size of each rule $p(|R||\eta_R)$ are modeled as Poisson distributions with means $\eta_{\mathcal{R}}$ and η_R . The probability of each rule is modeled as a weighted average

$$p(R_A|\boldsymbol{\rho}_A, \eta_r) = \sum_{r \in R_A, \rho \in \boldsymbol{\rho}_A} \rho p(|r||\eta_r), \quad (4.10)$$

over the probabilities of the corresponding productions. The weighting is given by the grammar parameters $\rho \in \boldsymbol{\rho}_A$ and the probability of each production corresponds to the Poisson distribution over its length $p(|r||\eta_r)$, given a desired production length η_r . Since all components are defined as discrete probabilities, the prior is always $p(\mathcal{G}) \leq 1$, eliminating the need for hard to tune weighting terms to cope with difficult scaling properties. Furthermore, the prior $p(\mathcal{G})$ will now prefer grammars with η_R productions per nonterminal with an average length of η_r symbols per production. The hyper-parameters η_R, η_r can be set to achieve a desired simplicity of the grammar. By weighting each production $r \in R_A$ with the corresponding grammar parameter $\rho \in \boldsymbol{\rho}_A$ the prior gives more significance to production which are more likely to occur.

4.5.2 Traversing the Grammar Space \mathfrak{G}

To find the optimal grammar \mathcal{G}^* , it is necessary to define mechanisms that generate new grammars. A common choice is to define operators $\text{op} \in \mathcal{O}$, where \mathcal{O} denotes the set of all operators. Each operator op manipulates the rule set \mathcal{R} and consequentially the nonterminal set \mathcal{V} of a given grammar \mathcal{G} , therefore, creating a new grammar \mathcal{G}' . For each operator op we define a domain Ω_{op} that op can act upon. The elements in Ω_{op} depend on the operator itself and can be for instance nonterminals, pairs of nonterminals or productions.

Each grammar represents a node in a grammar space \mathfrak{G} . The operators $\text{op} \in \mathcal{O}$ represent directed edges in \mathfrak{G} between two grammars. The grammar space \mathfrak{G} is illustrated in Figure 4.2. After grammar \mathcal{G}' was created by applying an operator op on grammar \mathcal{G} , the grammar parameters usually have to be recomputed. In this work, the parameters are re-estimated for every new grammar \mathcal{G}' via the Inside-Outside algorithm (Baker, 1979).

Not every possible grammar \mathcal{G} is suitable for sequencing movement primitives. Every sequence produced by \mathcal{G} has to guarantee a smooth, continuous trajectory within the state space of the MPs. In general, this means that a possible next primitive has to begin close to the end of the preceding primitive.

We restrict the grammar space \mathfrak{G} to only contain grammars that fulfill this connectivity requirement. The restriction is achieved by limiting the domain Ω_{op} of

each operator $\text{op} \in \mathcal{O}$, such that if grammar \mathcal{G} fulfills the connectivity requirement any grammar \mathcal{G}' resulting from an application of op on \mathcal{G} also fulfills the requirement. We incorporate the connectivity requirement into the definition of the two common operators merge and split.

The split Operator

The split operator divides the nonterminal $A_i \in \Omega_{\text{split}}$ into two new nonterminals A_j, A_k . The productions R_{A_i} are separated randomly into two corresponding, disjoint sets R_{A_j} and R_{A_k} , where none of the two resulting sets is empty. Each occurrence of A_i is randomly replaced by either A_j or A_k , where both A_j and A_k have to be selected at least once. The domain Ω_{split} contains all nonterminals with at least two productions. Furthermore, every nonterminal in Ω_{split} has to occur at least twice across all productions, including its own.

The merge Operator

The merge operator is the inverse operation to split and, hence, combines two nonterminals $(A_j, A_k) \in \Omega_{\text{merge}}$ into a new nonterminal A_i . Correspondingly, the productions of A_i are defined as the union $R_{A_i} = R_{A_j} \cup R_{A_k}$. Every occurrence of A_j and A_k is replaced by A_i . If A_j and A_k contain productions that begin or end in very different MP state spaces a merging would endanger the connectivity requirement. We avoid this problem by restricting the domain Ω_{merge} to only contain compatible nonterminal pairs. Assuming the sets $\text{first}(A)$ and $\text{last}(A)$ contain all possible primitives that could be at first or last position of any sequence produced starting from A . Two nonterminals A_j and A_k are now considered compatible if

$$\bigcup_{\theta \in \text{first}(A_j)} \text{c}\overleftarrow{\text{dn}}(\theta) = \bigcup_{\theta \in \text{first}(A_k)} \text{c}\overleftarrow{\text{dn}}(\theta) \quad \text{and} \quad \bigcup_{\theta \in \text{last}(A_j)} \text{c}\overrightarrow{\text{dn}}(\theta) = \bigcup_{\theta \in \text{last}(A_k)} \text{c}\overrightarrow{\text{dn}}(\theta).$$

The split and merge operators negate each other and are capable of generalizing existing hierarchies in grammars, however they lack the important ability to create new hierarchies.

The chunk Operator

In order to explore new hierarchies we additionally utilize the chunk operator (Stolcke, 1994), that creates a new nonterminal A with productions $R_A = \{r\}, r \in (\mathcal{A} \cup \mathcal{V})^+ \wedge r \in \Omega_{\text{chunk}}$. Every occurrence of the sequence r in a production in \mathcal{R} is replaced by A . The domain Ω_{chunk} contains all possible subsequences of all productions in \mathcal{R} .

The insert Operator

Finally, we define the new insert operator that negates the effects of chunk, by selecting a nonterminal $A \in \Omega_{\text{insert}}$ and replacing each occurrence of A with its production $r \in R_A$. The domain Ω_{insert} contains all nonterminals with exactly one production.

Given these four operators, we define the set of all possible operators as $\mathcal{O} = \{\text{merge, split, chunk, insert}\}$. Furthermore, the operators in \mathcal{O} are not exchangeable i.e., if a grammar \mathcal{G}' was created by applying the operator op on grammar \mathcal{G} , there exists no operator in $\mathcal{O} \setminus \{\text{op}\}$ that is able to produce \mathcal{G}' from \mathcal{G} .

4.5.3 Finding \mathcal{G}^*

Similarly to (Talton et al., 2012) we search for the optimal grammar $\mathcal{G}^* = \arg \max_{\mathcal{G}} p(\mathcal{G}|\mathcal{D})$ using Markov chain Monte Carlo (MCMC) optimization. A main advantage of MCMC over local search methods is that it's stochastic exploration traverses the grammar space better than local search methods. Given the definition of the grammar score the corresponding landscape is highly multimodal. Often several operators that each lead to a lower scoring grammar are required to be executed sequentially in order to arrive at a new maximum. Even with a broad beam width, beam search often fails to surpass such valleys whereas MCMC due to it's stochasticity manages to reach at least better local optima and even offers theoretical guarantees to find the global optimum in the limit.

In (Talton et al., 2012) the inputs are expected to already be hierarchical, restricting the grammar search to a reorganization of already existing productions by applying solely the merge and split operators. Given that our inputs are flat sequences, that is, pure sequences without hierarchy, of observed primitive samples, we additionally apply the chunk operator, that is capable of creating hierarchies (Stolcke, 1994). The insert operator ensures the irreducibility of the Markov chain. Analogously to (Talton et al., 2012), we apply the Metropolis Hastings algorithm. However, since (Talton et al., 2012) solely uses the split and merge operator, the paper directly defines the proposal distributions $q(\mathcal{G}'|\mathcal{G})$ as the probability of a split or a merge. In this work we define the proposal distribution as a mixture over the four operators $\mathcal{O} = \{\text{merge, split, chunk, insert}\}$,

$$q(\mathcal{G}', \text{op}'|\mathcal{G}) = \sum_{\text{op} \in \mathcal{O}} p(\text{op}'|\mathcal{G}, \eta_{\text{op}'}) q_{\text{op}}(\mathcal{G}'|\mathcal{G}, \text{op}'),$$

with mixture components $q_{\text{op}}(\mathcal{G}'|\mathcal{G}, \text{op}')$. The mixture probability is defined as

$$p(\text{op}'|\mathcal{G}, \eta_{\text{op}'}) = \frac{\eta_{\text{op}'} \left(1 - \delta_{|\Omega_{\text{op}'|}\right)}{\sum_{\text{op} \in \mathcal{O}} \eta_{\text{op}} \left(1 - \delta_{|\Omega_{\text{op}}|}\right)} \quad (4.11)$$

where $\eta_{\text{op}} \in \mathbb{R}$ is a weighting for the operator op , $\delta_{|\Omega_{\text{op}}|}$ denotes the Kronecker delta over the size of the domain Ω_{op} for operator op . Given that the operators in \mathcal{O} are not exchangeable, a mixture component $q_{\text{op}}(\mathcal{G}'|\mathcal{G}, \text{op}')$ should not contribute any probability mass if $\text{op} \neq \text{op}'$. This restriction is achieved by the Kronecker deltas $\delta_{\text{op}', \text{op}}$ in the following mixture components.

The split Probability $q_{\text{split}}(\mathcal{G}'|\mathcal{G}, \text{op}')$

Given that the split operator was applied to produce \mathcal{G}' from \mathcal{G} , there exist $A_i \in \mathcal{V}$ and $A_j, A_k \in \mathcal{V}'$. The chance of randomly selecting $A_i \in \Omega_{\text{split}}$ is $1/|\Omega_{\text{split}}|$. Additionally,

every production $r \in R_{A_i}$ was randomly assigned to either R_{A_j} or R_{A_k} , while each of those two sets had to be selected at least once. There are exactly $2^{|R_{A_i}|} - 2$ possibilities of assigning the productions to either R_{A_j} or R_{A_k} . Finally, the N_{A_i} occurrences of A_i across all productions in \mathcal{R} have been replaced by A_j or A_k in \mathcal{R}' . The chosen replacements have been one out of a total of $2^{|N_{A_i}|} - 2$ possibilities, considering that A_j and A_k had to be chosen at least once. Combining the possibilities for assigning the productions and for assigning the occurrences results in redundancies, since there are always two combinations that will result in the same \mathcal{R}' . Taking into account these redundancies yields the overall probability

$$q_{\text{split}}(\mathcal{G}' | \mathcal{G}, \text{op}') = \frac{\delta_{\text{op}', \text{split}}}{|\Omega_{\text{split}}|} \frac{2}{(2^{|R_{A_i}|} - 2)(2^{|N_{A_i}|} - 2)} \quad (4.12)$$

of \mathcal{G}' being produced from \mathcal{G} by using a split operator.

The merge Probability $q_{\text{merge}}(\mathcal{G}' | \mathcal{G}, \text{op}')$

The only stochasticity in the merge operator is the decision which pair $(A_i, A_j) \in \Omega_{\text{op}}$ is selected, yielding the simple probability

$$q_{\text{merge}}(\mathcal{G}' | \mathcal{G}, \text{op}') = \frac{\delta_{\text{op}', \text{merge}}}{|\Omega_{\text{merge}}|}. \quad (4.13)$$

that \mathcal{G}' is a result of a merge operation applied to \mathcal{G} .

The chunk Probability $q_{\text{chunk}}(\mathcal{G}' | \mathcal{G}, \text{op}')$

Given that the domain Ω_{chunk} already contains all possible subsequences of all productions in \mathcal{R} , the probability of \mathcal{G}' resulting from a chunk operation on \mathcal{G} is

$$q_{\text{chunk}}(\mathcal{G}' | \mathcal{G}, \text{op}') = \frac{\delta_{\text{op}', \text{chunk}}}{|\Omega_{\text{chunk}}|}, \quad (4.14)$$

which corresponds to simply choosing a particular sequence from Ω_{chunk} at random.

The insert Probability $q_{\text{insert}}(\mathcal{G}' | \mathcal{G}, \text{op}')$

The domain Ω_{insert} is already restricted to nonterminals with a single production. Selecting a nonterminal at random yields the probability

$$q_{\text{insert}}(\mathcal{G}' | \mathcal{G}, \text{op}') = \frac{\delta_{\text{op}', \text{insert}}}{|\Omega_{\text{insert}}|} \quad (4.15)$$

of \mathcal{G}' resulting from an insert operation on \mathcal{G} .

At every iteration of the Metropolis-Hasting algorithm a random new grammar is sampled from the proposal distribution $\mathcal{G}', \text{op}' \sim q(\mathcal{G}', \text{op}' | \mathcal{G})$. This new grammar is then accepted with a probability of

$$\text{acc}(\mathcal{G}', \text{op}' | \mathcal{G}) = \min \left(1, \frac{p(\mathcal{G}' | \mathcal{D})^{1/T} q(\mathcal{G}, \overline{\text{op}'} | \mathcal{G}')}{p(\mathcal{G} | \mathcal{D})^{1/T} q(\mathcal{G}', \text{op}' | \mathcal{G})} \right), \quad (4.16)$$

where T denotes a decaying temperature and $\overline{\text{op}'}$ denotes the complementary operator to op' , i.e., $\overline{\text{split}} = \text{merge}$, $\overline{\text{chunk}} = \text{insert}$. If the new grammar was accepted it is set to the current grammar $G \leftarrow G'$ and the next iteration begins. After a defined number of iterations, the grammar with the highest posterior is returned. For instance, Grammar 4.7 shows a grammar induced by the presented method given sequences of the previously described tic-tac-toe task. The semantically meaningful names of the nonterminals were chosen manually.

Given that the MCMC optimization finds high scoring grammar after only few iterations, the hyper-parameter optimization is inexpensive. Furthermore, a good rule of thumb for the number of productions per nonterminal and the number of symbols per production are 2 and 3 respectively, leaving the number of nonterminals the only free parameter of the presented prior.

4.6 Enhancing PCFGs with Attributes for Movement Primitive Sequencing

So far, the presented approach induces grammars, that do not violate the connectivity requirement. However, connectivity as defined in this work only guarantees that the transition area of two consecutive primitives is large enough to produce a continuous state space trajectory. In order to ensure smooth trajectories the start of the subsequent primitive has to be conditioned to the end of the current primitive. This can be achieved within the grammar formulation by introducing attributes. Furthermore, attributes can be used for defining points of interest that primitives need to reach for a successful execution. We introduce an evaluation scheme for movement primitive sequencing tasks that enhance given probabilistic context-free grammars with attributes and conditions. The scheme generalizes to different movement primitive sequencing tasks and, therefore, needs only little to no adaptation for specific tasks, with the exception of the initialization of the task-specific attribute values.

We define the following three attributes, common to primitive sequencing tasks:

- **transition** : This attribute defines where the current primitive ends and the next primitive is supposed to start. It is solely defined for nonterminals, and ensures that the produced primitives result in a continuous state space trajectory.
- **endpoint** : The endpoint of a movement primitive. It is solely defined for terminals and after the terminal has been evaluated, the **transition** attribute of the left hand side nonterminal is set to the **endpoint** of the corresponding primitive.

- `viapoints` : An ordered list of points that are supposed to be traversed by the sequence of primitives. The points are given in the state-space of the primitives. Once the first point is traversed by a primitive it is removed from the list and the next point is considered.

In addition to the attributes we define two conditions for necessary for the evaluation scheme. If preceded with an `assert` these conditions have to be satisfied for a successful evaluation.

- `reachable` : Given a primitive and a point in the primitive state-space, this condition is satisfied if the point is reachable by the primitive. In this thesis we use probabilistic movement primitives over the joint configuration of the robot. A point given in the configuration of the robot is reachable by a particular primitive if it is within two times the standard deviation of the trajectory mean of the movement primitive.
- `producible` : Given a nonterminal this condition is satisfied if at least one of the corresponding right-hand sides is producible. A right-hand side is considered producible if all mandatory conditions are satisfied, given the current set of attributes.

The described attributes enhance context-free grammars for movement primitive sequencing tasks, such that the sequenced primitives can be conditioned to state of the environment, e.g., the pose of an object. The conditions ensure a continuous state space trajectory of the sequenced primitives, even in the case of primitive adaptations.

4.6.1 Evaluation Scheme for the Tic-Tac-Toe Task

We explain the functionality of the attributes in detail using the example of the tic-tac-toe task. We start with the probabilistic context-free grammar shown in Grammar 4.7. The grammar was induced from demonstrations as described in the previous section.

The production of the sequence always begins at the `START` nonterminal. We assign two points to the `viapoints` attribute. One for the position of a stone and one for the field the stone is supposed to be placed on. Furthermore, we set the `transition` attribute to the current position of the robot in the primitive state-space. We use the literals `stone_pos`, `field_pos` and `cur_pos` instead of the actual numerical values

```

START.transition = cur_pos
START.viapoints = [stone_pos, field_pos]
assert: producible(START)
assert: empty(START.viapoints)

START  →  MOVE                                (1.00)                (GR 4.2)
        MOVE.viapoints = START.viapoints
        MOVE.transition = START.transition
        assert: producible(MOVE)
        START.transition = MOVE.transition
        START.viapoints = MOVE.viapoints .

```


The `assert` attribute indicates that this condition must be satisfied otherwise the entire right-hand side is removed from consideration as a possible production of the corresponding nonterminal given the current attribute set. If the `START` nonterminal is not producible, the task is not solvable under the given attributes. Furthermore, if the `viapoints`-list is not empty after evaluating `START` not all points were traversed and the task is not considered solved.

An important convention in the attribute notation is that whenever a nonterminal appears as an argument of a condition or on the right-hand side of an assignment it has been evaluated before. For instance, the producibility of `START` and `MOVE` can only be asserted once the respective nonterminal has been fully evaluated.

The `MOVE` nonterminal contains multiple productions, each consisting of multiple symbols. The productions can be evaluated in parallel, i.e., the evaluation of each of the productions begins with the same set of attributes, independent of the changes that have occurred during the evaluation of the other productions. In contrast, the symbols of a single production are evaluated sequentially, i.e., every symbol begins with the attributes set after the evaluation of the previous symbol. As mentioned before, terminals represent single movement primitives. It is important that a sequence of primitives does not contain any jumps in the state-space, since a real robot platform will not be able to make significant changes in its configuration instantaneously. Therefore, we ensure that every selected primitive starts where the previous primitive ended. In the proposed evaluation scheme this is achieved by ensuring that the `reachable` condition holds for the primitive and the current `transition-point`. If the primitive can start from the `transition-point`, the `transition` attribute is set to the endpoint of the primitive afterwards. Furthermore, we define a function to traverse the `viapoint`-list.

- `traverse` : The function expects a terminal and a list of points. If the first point in the list is `reachable` by the terminal the corresponding primitive will traverse the point, the point will be removed from the list and the function evaluates to `true`.

Given that the possible adaptation of the primitive to the point could change the endpoint, `traverse` has to be evaluated before the `transition-point` is adapted

```

MOVE  →  pick_near TO                                (0.40)
        assert: reachable(pick_near, MOVE.transition)
        traverse(pick_near, MOVE.viapoints)
        MOVE.transition = pick_near.endpoint
        TO.viapoints = MOVE.viapoints                (GR 4.3)
        TO.transition = MOVE.transition
        assert: producible(TO)
        MOVE.viapoints = TO.viapoints
        MOVE.transition = TO.transition .

```

Grammar 4.3 only shows the evaluation for one of the two productions. The evaluation of the other production is defined analogously, but with the terminal `pick_far`

instead of `pick_near`. Despite that both of the productions next evaluate the TO non-terminal

```
TO  →  LEFT home                                (0.47)
      LEFT.viapoints = TO.viapoints
      LEFT.transition = TO.transition
      assert: producible(LEFT)
      TO.viapoints = LEFT.viapoints              (GR 4.4)
      TO.transition = LEFT.transition
      assert: reachable(pick_near, TO.transition)
      traverse(pick_near, TO.viapoints)
      TO.transition = pick_near.endpoint ,
```

the actual evaluations might differ due to two different sets of attribute values. Again two different possible productions are evaluated in parallel but only one is shown. The evaluation of the other production is defined equivalently, but with the nonterminal RIGHT instead of the LEFT. In contrast to the evaluation of MOVE the productions of TO require the evaluation of a nonterminal before the evaluation of a terminal.

4.6.2 A General Evaluation Scheme for Sequencing Tasks

A pattern for both terminal and nonterminal evaluations is clearly evident. Every terminal `a` on the production of a rule with nonterminal `A` on the left-hand side is evaluated using

```
assert: reachable(a, A.transition)
traverse(a, A.viapoints)
A.transition = a.endpoint
```

and every nonterminal `B` on the right-hand side of a rule with nonterminal `A` on the left-hand side is evaluated using

```
B.viapoints = A.viapoints
B.transition = A.transition
assert: producible(B)
A.viapoints = B.viapoints
A.transition = B.transition .
```

The presented evaluation scheme is very general and can be applied to any movement primitive sequencing task. Using not further specified via-points has the advantage that the evaluation does not restrict which primitive traverses which point. For instance, in the case of an obstacle it might be sufficient that the obstacle is passed at some point, but it does not necessarily matter which primitive avoids it. However, the unspecified list of via-points has a significant disadvantage. A primitive might require a certain via-point, for instance `pick_near` and `pick_far` have to know where the stone is positioned in order to pick it up successfully. Nothing in the current scheme associates via-points with a certain primitives. We solve this problem by introducing two additional attributes.

- **keywords** : An unordered list of keywords. This attribute is assigned only to terminals before the evaluation and contains keywords identifying relevant points in the **targets** attribute.
- **targets** : A dictionary that maps keywords to ordered lists of points. The points are defined in the primitive state-space. A primitive containing a matching keyword in its **keywords** attribute extracts the first point in the corresponding list.

The evaluation scheme for terminals is now defined as

```

assert: reachable(a, A.transition)
for: key in a.keywords
    assert: key in A.targets
    assert: traverse(a, A.target[key][0])
traverse(a, A.viapoints)
A.transition = a.endpoint ,

```

(GR 4.5)

where the **for:** notation indicates an iteration and the **in** notation indicates the existence of an element in the respective list. The **targets** attribute can strongly influence the production of a sequence. The given target could be outside of the distribution of the primitive associated with the terminal. For instance, both terminals **pick_near** and **pick_far** have a **stone** keyword. If the **targets** attribute associates **stone** with a value outside of the **pick_near** primitive but within the **pick_far** primitive, the **assert** statement would only hold for **pick_far**, ensuring that every sequence produced with this set of targets will contain a **pick_far** and never a **pick_near**. This way the target attributes directly influence the effective structure of the grammar.

The evaluation scheme for nonterminals only changes such that the **targets** attribute is additionally passed down and received afterwards, analogously to the **viapoints** attribute.

4.6.3 Evaluating Parallel Attribute Sets

We already established that the right-hand sides of a single nonterminal are evaluated in parallel. If more than one right-hand side does not violate any asserts, multiple parallel sets of attributes return from that nonterminal evaluation. Given that within one right-hand side the attributes are passed sequentially from symbol to symbol, the question arises which of the multiple attribute sets should be considered. A naive approach would be to select a random attribute set. However, one attribute set might result in an unproducible right-hand side while another might not. We address this problem by storing every attribute set corresponding to a producible right-hand side in an ordered list. The order is defined randomly, while being weighted with the probabilities of the right-hand sides. Only the first set of attributes is considered, unless the set results in an assert violation, then the attribute set is discarded and the evaluation continues with the next set in the list. If no sets are left, the right-hand side is considered unproducible. It is possible that a given set of targets results in an effective grammar structure that is not capable of producing any sequence of primitives. For instance, neither **place_left** nor **place_right** are able to place the stone

outside of the playing field. Hence, if the corresponding target is set outside the playing field neither of the productions of the TO terminal will be producible and the non-producibility will be propagated up until the start symbol. In this case the grammar would return an empty sequence. This can easily be used to prompt the user that the current grammar can not produce a sequence satisfying the given set of targets. Therefore, different targets or new demonstrations extending the grammar are required.

The presented attributes and evaluation scheme are independent of the actual task itself and generalize over movement primitive sequencing tasks. The only attribute that has to be accessed and potentially adapted by the user are the targets. Hence, the remaining attributes and the evaluation scheme itself can be considered constants and can be hidden from the user, concealing necessary complexity that does not affect the representation of the behavior. We further simplify the presentation of the attribute grammar, by presenting the keywords of the targets attribute as grammar attributes themselves. By applying these simplifications we arrive at Grammar 4.1 as presented in the problem statement.

4.7 Experiments

We evaluated the proposed approach on several real robot tasks. First, we induced a grammar producing turns of the tic-tac-toe game. Second, we learned a grammar that assists a human with the assembly of a simple toolbox. In both tasks the necessary primitives were encoded as Probabilistic Movement Primitives (Paraschos et al., 2017a). For each of the tasks we compare the posterior resulting from our proposed prior, *Grammar Poisson*, with the one resulting from three common structure prior choices, *MDL*, *Poisson + MDL*, *Avg. Poisson*. The *MDL* prior is simply defined as an exponential distribution with the MDL as its energy (Talton et al., 2012). The *Poisson + MDL* prior weights the description language for every production with the Poisson probability over the length of the production (Kitani et al., 2008). Finally, the *Avg. Poisson* prior discards the MDL completely and is solely represented by a Poisson distribution over the average length of all productions (Lee et al., 2013). A major difference of the *Grammar Poisson* prior to the other discussed priors is that we do not model the distribution over the grammar parameters as a Dirichlet distribution but rather use them as a weighting for the average production length.

4.7.1 Learning a Grammar for Tic-Tac-Toe Turns

In this task we learned a grammar that allows the robot to play tic-tac-toe against a human. Each produced sequence corresponds to one turn of the game, i.e., picking a stone, closing the hand, placing the stone on the field, opening the hand and returning to the home position. The goal is not to learn the logic behind the game but rather the induction of an intuitive grammar producing valid turns. The segmentation of the demonstrations and, hence, the learning of the primitives was done beforehand via Probabilistic Segmentation (Lioutikov et al., 2017a). The five resulting arm primitives are shown in Figure 4.5, where the green and blue highlighted areas mark the start

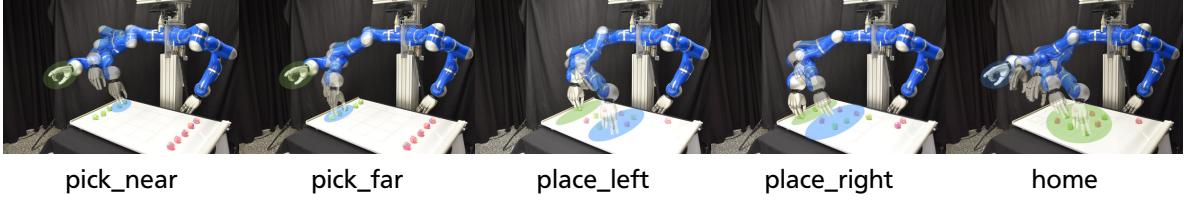


Figure 4.5.: The five arm primitives used in the sequences, representing turns in the tic-tac-toe game. While both `pick_near` and `pick_far` approach a stone from the home position they differ in the stone positions they can reach. Similarly the primitives `place_left` and `place_right` position the stone in different areas of the playing field.

and end of the end-effector. While `pick_near` and `pick_far` are semantically similar, they actually differ quite substantially in the encoded joint trajectory of the robot and, hence, the segmentation algorithm separated those movements into two separate primitives. The same explanation holds for `place_left` and `place_right`.

The grammar induction method was given 15 observations of four unique sequences, each consisting of five terminals

START	→	DEMO1 (0.33)		DEMO2 (0.20)	
	→	DEMO3 (0.27)		DEMO4 (0.20)	
DEMO1	→	pick_far close place_right open home		(1.00)	
DEMO2	→	pick_near close place_right open home		(1.00)	(GR 4.6)
DEMO3	→	pick_far close place_left open home		(1.00)	
DEMO4	→	pick_near close place_left open home		(1.00).	

We initialized our approach with a desired number of rules $\eta_{\mathcal{R}} = 5$, the desired number of average productions per rule $\eta_{\mathcal{R}} = 2$ and the desired average length of each production $\eta_r = 3$. The weights for each operator were set uniformly to $\eta_{op} = 1$, $op \in \mathcal{O}$. The MCMC optimization was run for 400 steps and resulted in 324 accepted grammars. The corresponding normalized posteriors are shown in Figure 4.6. The grammar at index 171

START	→	MOVE (1.00)			
MOVE	→	pick_near TO (0.40)		pick_far TO (0.60)	
TO	→	LEFT home (0.47)		RIGHT home (0.53)	(GR 4.7)
LEFT	→	close place_left open		(1.00)	
RIGHT	→	close place_right open		(1.00)	

produces the highest posterior. The induced Grammar 4.7 intuitively represents that each produced sequence will move a near or a far stone to either the left or the right side of the playing field. Furthermore, after every closing of the hand there will be a later opening of the hand. A possible sequence produced by the grammar, including the corresponding parse tree is seen in Figure 4.7. The parse tree includes keys and values assigned to the keywords and targets attributes. The production of the sequence was started with the attribute `targets = {stone : stone_pos, field :`

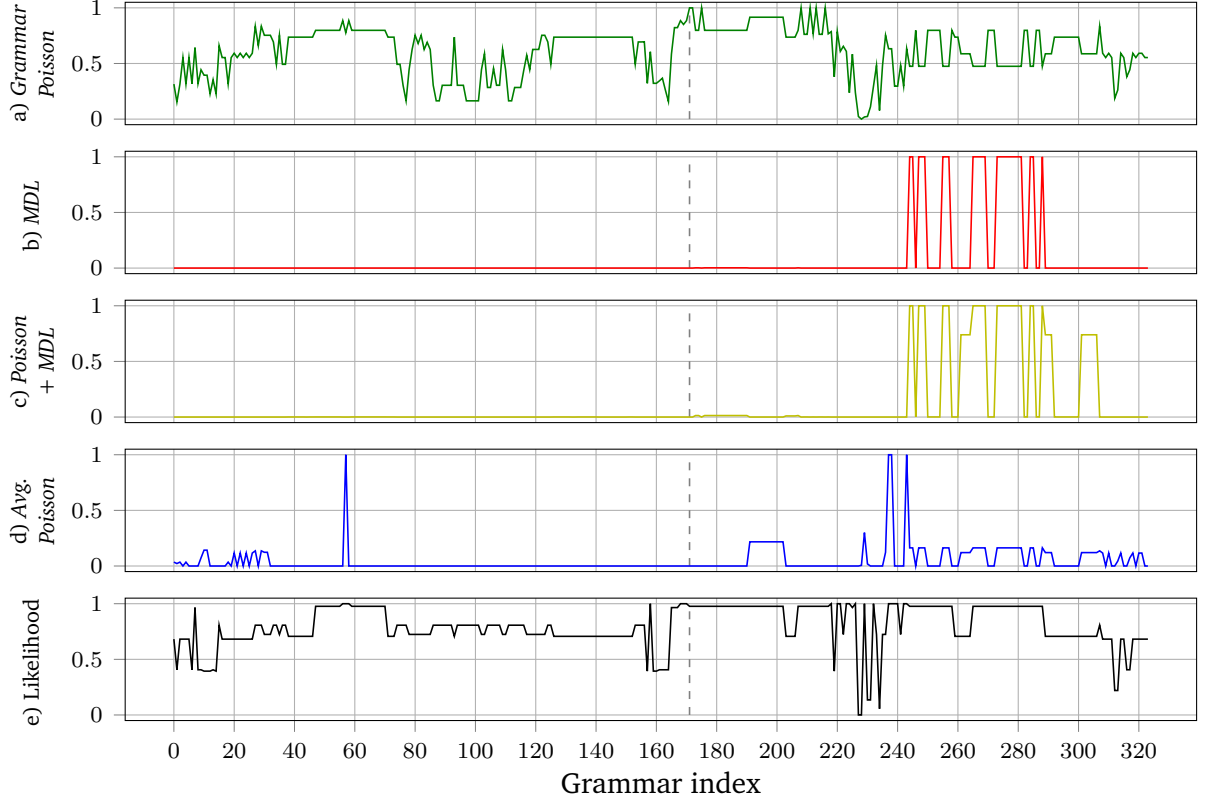


Figure 4.6.: The posteriors and the likelihood for the tic-tac-toe turn grammar. The vertical, dashed line indicates the index of the highest posterior (171), given the presented Poisson prior.

field_pos} consisting of the position of the stone that should be played next, stone_pos and the field position field_pos on which the stone should be placed. For simplicity, the parse tree presented to the user replaces the actual position of stone_pos and field_pos but instead the numbering of the corresponding playing field cell.

The naming of the nonterminals was chosen manually after the grammar learning. An automated naming of the nonterminals corresponding to the semantics of the productions is outside of the scope of this thesis and remains part of future work. Figure 4.6b-d shows the normalized posteriors corresponding to the three common priors. The x-axis corresponds to the different grammars traversed during the MCMC optimization, i.e., the grammar $\mathcal{G}^i, \text{op}^i \sim q(\mathcal{G}^i, \text{op}^i | \mathcal{G}^{i-1})$ was sampled from the proposal distribution around \mathcal{G}^{i-1} by applying op^i . The spiky behavior of the posteriors (b-d) is due to the uninformative Dirichlet prior for the grammar parameters and the exponential distribution over the *MDL*. Both of these factors can change significantly with a small change in the grammar, e.g., a merge creating a rule with many productions or a chunk reducing the length of a long production. Furthermore, it is noticeable that the likelihood of the grammar $p(\mathcal{G}^i | \mathcal{D})$ does not play significantly into the posteriors of (b-d), whereas our posterior (a) shows a much stronger dependency on the likelihood. This behaviour, is explained by the fact that the likelihood as introduced in Equation 4.5 is a probability mass function, but the three priors (*MDL*, *Poisson + MDL*, *Avg. Poisson*) are products of probability density functions. In contrast, our prior (*Grammar Poisson*) is defined as a probability mass function, averaging

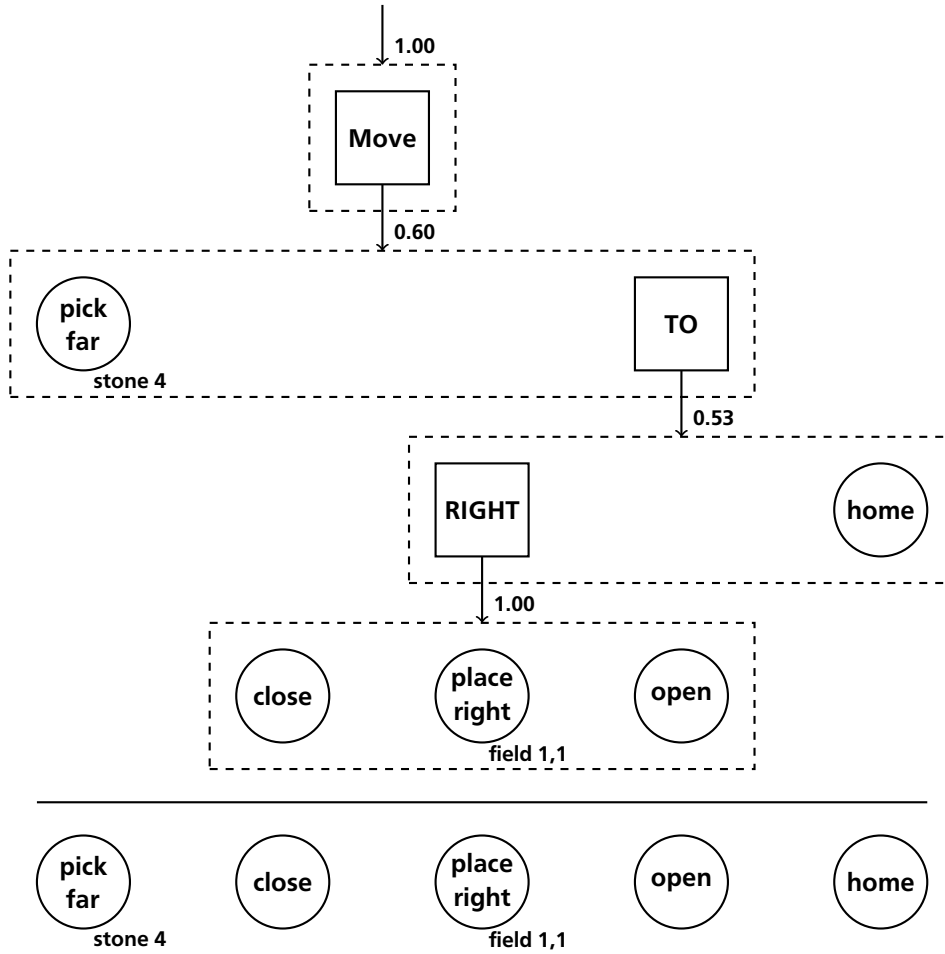


Figure 4.7.: A parse tree of a sequence produced by the learned grammar for tic-tac-toe turns. Nonterminals are presented as squares and terminals as circles. A dashed rectangle represents the production chosen by the parent terminal with the probability next to the connecting arrow. The solid line separates the final sequence from the producing parse tree. The grammar was enhanced with the presented attributes and evaluation scheme, where stone and field are two keys assigned to the keywords attributes of the `pick_far` and `place_right` terminals respectively.

over multiple Poisson distributions. This definition prohibits the prior from completely dominating the likelihood. As a consequence, the proposed prior (*Grammar Poisson*) results in a posterior (a) that takes the given observations much stronger into account than the posteriors in (b-d).

4.7.2 Learning a Grammar for a Simple Toolbox Assembly

This task shows the abstraction capabilities of our approach. The demonstrations were again segmented beforehand and resulted in the five arm primitives, shown in Figure 4.8, and four hand primitives, closing and opening the hand for both a board and a screw grasp. The set of demonstrations contained three different sequences, consisting of 40 terminals each. Every observation showed the grasping and handing over of four boards and four screws, either alternating between the

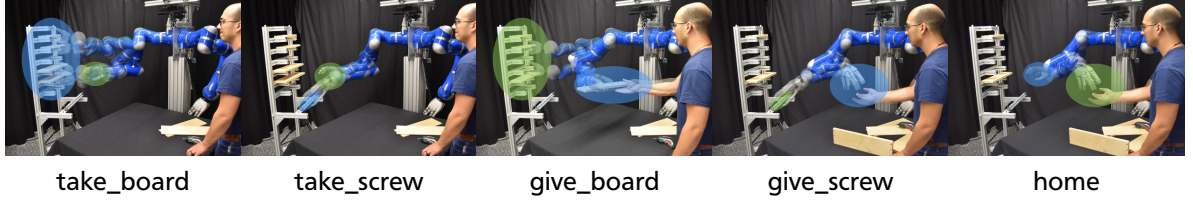


Figure 4.8.: The arm primitives of the box assembly task. The robot applies different primitives for grasping a board, `take_board`, or picking a screw, `take_screw`. Similarly the handover for boards and screws is encoded in different primitives.

board and the screw or starting with two boards and alternating subsequently. The approach was initialized with $\eta_{\mathcal{R}} = 9$, $\eta_{\mathcal{R}} = 2$, $\eta_r = 2$. The weights for the split and merge operators were set to 1 and the remaining two were set to 2. The MCMC optimization ran for 400 iterations and 303 grammars were accepted. The posteriors for the accepted grammars are shown alongside the likelihood in Figure 4.9. The posteriors show similar behavior as in the previous task. Both the *MDL* and the *Poisson + MDL* have a maximum at 162, indicating that the corresponding grammar has the minimal description length of all accepted grammars. The *Avg. Poisson* prior has its maximum at 44 due to an average production length close to η_r . However, the corresponding grammar contains 14 rules with one production each. The grammar with the maximum posterior according to the *Grammar Poisson* prior is given at index 160

START	→ ASSEMBLE_SB	(0.5)	
	→ ASSEMBLE_BB	(0.5)	
BOARD	→ take_board GIVE_B home	(1.0)	
SCREW	→ take_screw GIVE_S home	(1.0)	
BSB	→ BOARD SCREW BOARD	(1.0)	
SBS	→ SCREW BOARD SCREW	(1.0)	
GIVE_S	→ close_screw give_screw open_screw	(1.0)	(GR 4.8)
ASSEMBLE_BB	→ BOARD BOARD SBS ...		
	BOARD SCREW SCREW	(1.0)	
GIVE_B	→ close_board give_board open_board	(1.0)	
ASSEMBLE_SB	→ SBS BOARD SCREW BSB	(0.5)	
	→ BOARD SBS BOARD SBS	(0.5).	

The learned grammar abstracts a full turn from taking a board or screw until going back to the home position. This subsequence was not marked in any way and was detected as a consequence of the grammar learning. The sequence occurred multiple times during each observation. Abstracting it into a nonterminal will therefore simplify the grammar significantly. Furthermore, the grammar encodes that a grasping of a board or a screw through the closing of the hand has to be eventually followed by the corresponding opening of the hand. The alternation between handing over a board and a screw is represented in the two rules for *SBS* and *BSB* and the rules for

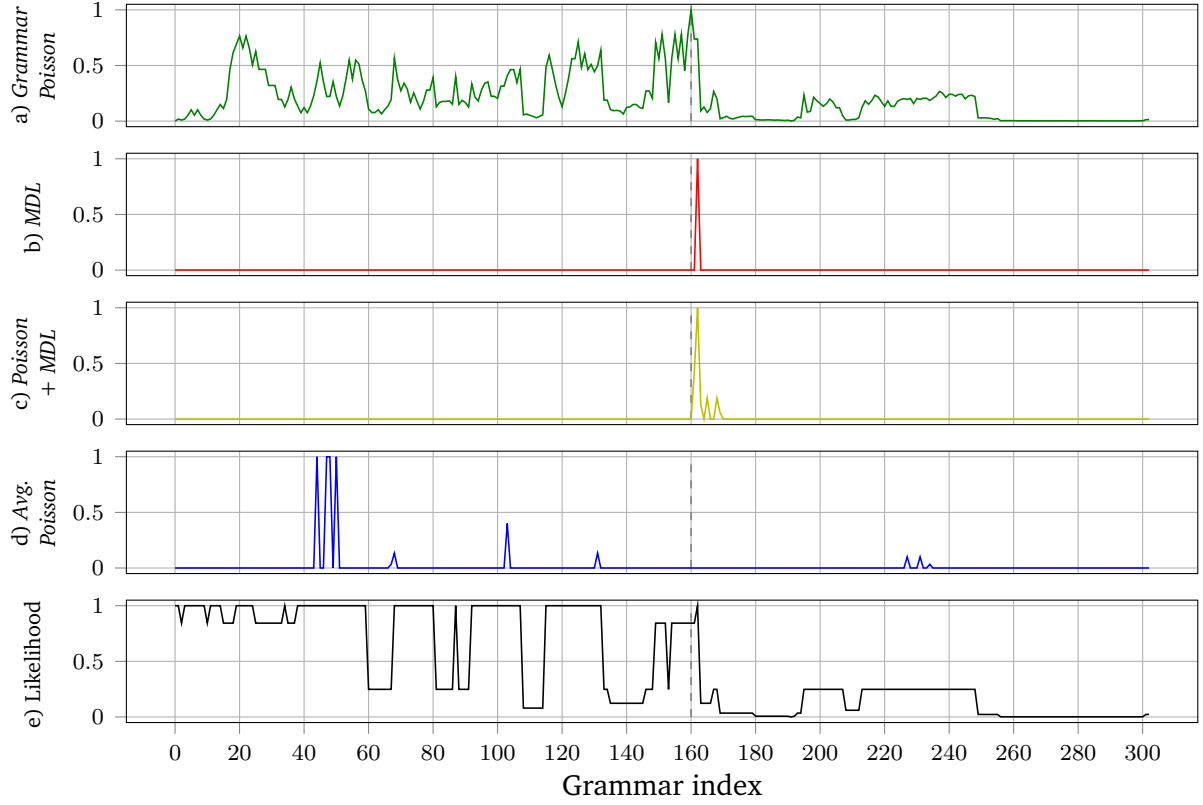


Figure 4.9.: The posteriors and the likelihood for the box assembly task. The vertical, dashed line indicates the index of the highest posterior (160), given the presented Poisson prior.

ASSEMBLE_SB. The option of starting with two boards is encoded in ASSEMBLE_BB. Grammar 4.8 is able to produce all observed alternations of the box assembly

```
SCREW BOARD SCREW BOARD SCREW BOARD SCREW BOARD
BOARD BOARD SCREW BOARD SCREW BOARD SCREW SCREW
BOARD SCREW BOARD SCREW BOARD SCREW BOARD SCREW ,
```

where SCREW and BOARD are the nonterminals abstracting the full subsequences of taking a board or screw and handing it over to the collaborator.

4.8 Conclusion

In this work, we have introduced attribute grammars as a mechanism to sequence movement primitives. We have shown how to identify the categories of movement primitives and how to determine if two probabilistic movement primitives are connectible or not. The presented categorization approach is simple yet efficient, however, in future work we want to investigate more sophisticated approaches for the clustering of parameterized time series such as the applied movement primitives. Furthermore, we have presented an approach that induces probabilistic context-free grammars from flat sequences of movement primitive samples, i.e., no hierarchy in the observations, while taking advantage of a stochastic primitive representation.

The novel grammar prior is defined over several coupled Poisson distributions, and eliminates the many complications that arise from both Dirichlet parameter priors and minimal description length based structure priors. In our method, the hyperparameters of the prior have a clear semantic interpretation, namely the number of productions for each nonterminal and the average length of each production. The posterior is learned using a Markov chain Monte Carlo optimization where the proposal distribution is formulated as a mixture model over four operators. We defined attributes and conditions of a general evaluation scheme for sequencing tasks. We enhanced an initially induced probabilistic context-free grammar for making a move in a game of tic-tac-toe with the defined attributes and the evaluation scheme. While the MCMC optimization is less likely to get stuck in local optima than other suggested search strategies, such as beam search, it is not without fault. Depending on the complexity of the task with respect to the length of the observed sequences and the number of terminals, a significant number of samples are required to reach a promising area of the search space. Given that, the actual interest of grammar induction is not the exploration of the posterior, but rather the finding of the optimal grammar inside the search space, future work, will investigate the advantages of Monte Carlo tree search over MCMC for this particular challenge. Another future line of research is the goal to learn more general grammars while avoiding an over generalization, effectively defying Gold’s Law. A possible approach is to take advantage of the grammar as a generative model and introduce reinforcement learning techniques to improve the grammar after it has been induced from a given set of demonstrations.



5 Reinforcement Learning for Formal Grammars

In the previous chapter we presented an approach for the induction of probabilistic context-free grammars from observed sequences. The grammar parameters were determined using the inside-outside algorithm (Baker, 1979). This EM based approach, learns the parameters based on the given observations. The probabilities learned by that approach best explain the given data without taking mistakes or any type of quality of the observations into consideration. Given that learned grammars are commonly applied for the verification and identification of new incoming sequences, such as checking the syntax of statements in programming languages, this limitation

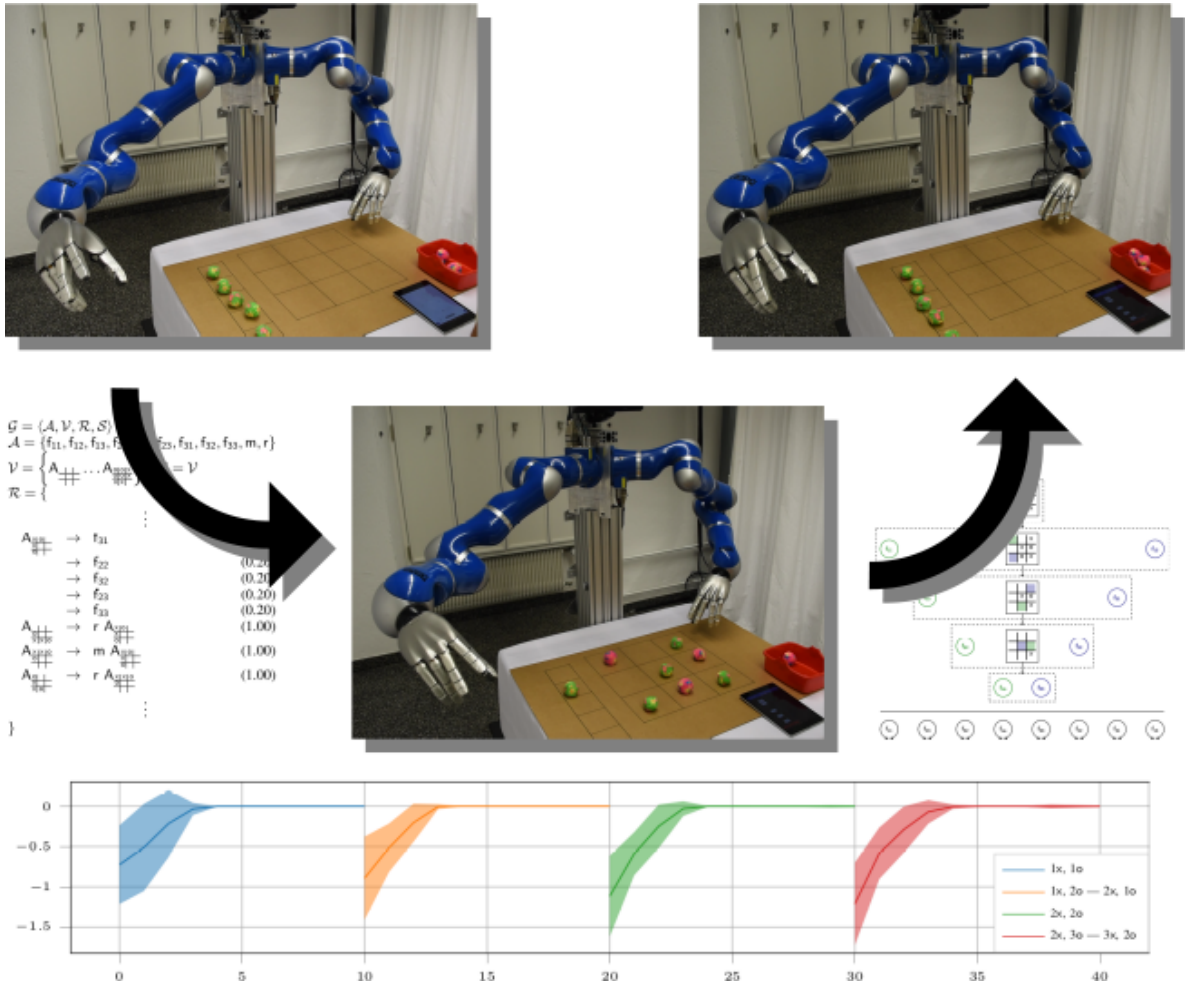


Figure 5.1.: In this chapter we formulate the natural policy gradient method for formal grammars. We learn the grammar parameters to learn a strategy for playing tic-tac-toe and a strategy to clean the board bi-manually while avoiding collisions between the arms.

is often accepted or even desired. Most algorithms for the grammar parameter estimation by themselves simply do not assess the quality of the observation during the parameter update and by default expect every observation to be positive. However, Gold’s rule states that in general no grammar can be correctly induced from positive examples only (Gold, 1967). Therefore, approaches have been proposed that require observations explicitly labeled as negative examples, to improve the grammar induction or the parameter update (Smith and Eisner, 2005). Recently Gold’s rule came under scrutiny since it appears that the rule is not backed up by empirical evidence as for instance in the learning behavior of children (Marcus, 1993) or the learning process of Songbirds (Ondracek and Hahnloser, 2013). In both examples, the amount of negative examples is not significant enough to explain the learning progress. In this chapter we introduce the idea that the improvement of a given grammar does not necessarily require explicit negative examples but rather an assessment of the quality of the samples produced by the grammar itself. This idea is of particular interest for scenarios where grammars are applied in a generative manner. For instance, in (Lioutikov et al., 2017b), grammars were used for sequencing movement primitives in order to place a stone in a game of tic-tac-toe and to hand over boards and screws in a simple collaborative box assembly task. Given suboptimal demonstrations or a change in the scenario, the observed sequences might not suffice to produce an optimal behavior of the robot. In such cases a parameter estimation based on the original demonstrations is no longer satisfactory anymore. We propose a novel approach to learn the grammar parameters based on a well known reinforcement learning approach. In particular, we formulate and apply the natural policy gradient method (Kakade, 2002) to formal grammars. We evaluate the approach on a simple grammar for the language $a^n b^n$ and apply the method to learn a strategy for playing tic-tac-toe as well as solving the task of cleaning up a tic-tac-toe playing field bi-manually without collision, as illustrated in Figure 5.1.

5.1 Natural Policy Gradient for Formal Grammars

In this section we explain how to apply the natural policy gradient method to formal grammars. Chapter 2 explained the general concept of grammars and parse trees. Following that introduction, the probability of an observed sentence \mathbf{a} given the grammar \mathcal{G} is defined as

$$\mathbf{a} \sim p(\mathbf{a} | \mathcal{G}_{\Theta}) = \sum_{T \in T(\mathbf{a})} p(T | \mathcal{G}_{\Theta}), \quad (5.1)$$

$$p(T | \mathcal{G}_{\Theta}) = \prod_{\alpha_{A,j} \in T} p_{\theta_A}(\alpha_{A,j}), \quad (5.2)$$

with $T(\mathbf{a})$ being the set of all parse trees producing \mathbf{a} . The probability $p_{\theta_A}(\alpha_{A,j})$ describes the probability of $\alpha_{A,j}$ being produced by the rule for nonterminal A under

the parametrization θ_A . Given that the probabilities of each rule sum up to one, $p_{\theta_A}(\alpha_{A,j})$ represents a the categorical distribution

$$p_{\theta_A}(\alpha_{A,j}) = \prod_i^{|A|-1} \theta_{A,i}^{\delta_{ij}} \left(1 - \sum_i^{|A|-1} \theta_{A,i} \right)^{\delta_{j|A|}}.$$

Using the probabilities ρ_A directly as parameters would yield an over-parameterized distribution. Instead, we opt for a parametrization where the last production is assigned the probability not accounted for by the preceding productions, ensuring a proper multinomial as long as the remaining parameters are well-behaved. The parameters for the entire grammar are now given as $\Theta = \{\theta_A | A \in \mathcal{V}\}$. The goal of this work, is to learn the parameters Θ such that the sentences $\mathbf{a} \sim \mathcal{G}_\Theta$ produced by the grammar maximize a given objective $J(\Theta)$. Given, that every parameter $\theta_{in}\Theta$ is part of a categorical distribution and, hence, a probability, it is important, that the learning process updates the parameters in small enough increments, to maintain the multinomial characteristic of the parametrization. A policy gradient approach, introduced specifically for such small increments independent of the parametrization of the policy is the natural gradient method, described next.

5.1.1 Grammar as Policy

A policy is generally described as a parameterized conditional distribution $\pi_\Theta(\mathbf{u}|\mathbf{x})$ with actions \mathbf{a} , state \mathbf{x} and parameters Θ . In this work we regard the sentences produced by the grammar as actions $\mathbf{u} = \mathbf{a}$ and define the state independent policy

$$\pi_\Theta(\mathbf{a}) = p(\mathbf{a} | \mathcal{G}_\Theta), \quad (5.3)$$

which is simply the probability shown in Equation 5.1. Note, that the grammar \mathcal{G} is not used as a state, since it influences the policy only through its structure, making it an additional parametrization rather than a state. Even if the grammar would be considered a state, the policy definition remains, since a produced sentence does not change the grammar itself. By not defining a state, the given formulation is open to the introduction of state variables. However, the scope of this work is restricted to the presented bandit setting.

As Equation 5.1 shows, a single sentence \mathbf{a} could be assigned to multiple parse trees $T \in T(\mathbf{a})$. While this ambiguity does not necessarily pose a problem, it is generally undesired. We deal with this potential ambiguity by not learning the policy over sentences but directly by learning the policy over parse trees

$$\pi_\Theta(T) = p(T | \mathcal{G}_\Theta). \quad (5.4)$$

For unambiguous grammars both policies are equivalent. For ambiguous grammars, however, the latter policy has a significant advantages. Assuming the same sentence \mathbf{a} achieves a good reward and can be explained by two separate parse trees T_{good} and T_{bad} . Basing the policy update purely on \mathbf{a} will strengthen the productions involved

in both parse trees equally, independent of the structures of the two parse trees or the productions involved. For instance, the policy update would ignore if T_{bad} is a degenerate tree and T_{good} is balanced tree or if the productions in T_{good} always lead to good results and the productions in T_{bad} are primarily involved in bad results. Sampling trees instead of sentences, allows to include the production of a sentence into the reward function. An analogy from behavioral psychology would be to not only analyze the behavior but also the motive.

5.1.2 Natural Policy Gradient for Grammars

Given a path of actions, or in our case trees, $\tau = [T_1 \dots T_H]$ with horizon H , policy gradient methods aim to optimize the policy parameters Θ iteratively, such that the expected return

$$J(\Theta) = \mathbb{E}_{\Theta} \left(\sum_{t=1}^H \beta_t r(T_t) \right) \quad (5.5)$$

is maximized, where $r(T_t)$ and β_t denote the reward and discount factor for time step t . At every iteration h the parameters are updated according to the gradient update rule

$$\Theta_{h+1} = \Theta_h + \gamma_h \tilde{\nabla}_{\Theta} J(\Theta_h), \quad (5.6)$$

with learning rate γ_i . Choosing the learning rate can be crucial for the success of the gradient method. We apply a separate learning rate for each parameter and adapt each learning rate at every iteration according to the simple Rprop scheme (Riedmiller and Braun, 1992). By applying the log trick, the gradient of the policy can be formulated as an expectation over the gradients of the log policies for each time step

$$\nabla_{\Theta} J(\Theta) \approx \left\langle \left(\sum_{t=1}^H \nabla_{\Theta} \log \pi_{\Theta}(T_t) \right) \left(\sum_{t=1}^H \beta_t r(T_t) - b \right) \right\rangle, \quad (5.7)$$

where $p(\tau)$ is the probability of the observed path τ and $\langle \cdot \rangle$ denotes the sample average. The constant baseline b reduces the variance of the gradient estimator. In this work, we simply use the average reward as baseline $b = \left\langle \left(\sum_{t=1}^H \beta_t r(T_t) \right) \right\rangle$

Given the Fisher-information matrix \mathbf{F} the natural policy gradient

$$\tilde{\nabla}_{\Theta} J(\Theta) = \mathbf{F}^{-1} \nabla_{\Theta} J(\Theta) \quad (5.8)$$

ensures small gradient updates (Kakade, 2002). This behavior is achieved by bounding the change between the path distributions $p_{\Theta_h}(\tau)$ and $p_{\Theta_{h+1}}(\tau)$.

The gradient for each tree $\nabla_{\Theta} \log \pi_{\Theta}(T)$ is defined over the first order partial derivatives for each nonterminal rule

$$\frac{\partial \log \pi_{\Theta}(T)}{\partial \theta_{B,i}} = \sum_{\alpha_{A,k} \in T} \delta_{AB} \frac{\partial \log p_{\theta_A}(\alpha_{A,k})}{\partial \theta_{A,i}}, \quad (5.9)$$

$$\frac{\partial \log p_{\theta_A}(\alpha_{A,k})}{\partial \theta_{A,i}} = \frac{\delta_{ki}}{\theta_{A,k}} - \frac{\delta_{k|A|}}{1 - \sum_{l=1}^{|A|-1} \theta_{A,l}} \quad (5.10)$$

with nonterminals A, B and Kronecker delta $\delta_{AB} = 1 \iff A = B$.

Similarly to the policy gradient, the Fisher-information matrix can also be estimated by an sample average, whereas the average is now computed over the log policy Hessians for each tree. The Fisher-information is now given as the negative of that sample average $\mathbf{F} = - \left\langle \sum_{t=1}^H \text{Hess} \log \pi_{\Theta}(\mathbf{T}_t) \right\rangle$. Hence, each element of the matrix

$$F_{ij} = - \left\langle \sum_{t=1}^H \frac{\partial^2 \log \pi_{\Theta}(\mathbf{T}_t)}{\partial \theta_{B,i} \partial \theta_{C,j}} \right\rangle \quad (5.11)$$

corresponds to the negative sample average over the second order partial derivatives

$$\frac{\partial^2 \log \pi_{\Theta}(\mathbf{T})}{\partial \theta_{B,i} \partial \theta_{C,j}} = \sum_{\alpha_{A,k} \in \mathbf{T}} \delta_{AB} \delta_{AC} \frac{\partial^2 \log p_{\theta_A}(\alpha_{A,k})}{\partial \theta_{A,i} \partial \theta_{A,j}}. \quad (5.12)$$

Given this formulation it is evident, that the Hessian for each tree is a block diagonal, where each block corresponds to the Hessian of a nonterminal rule with second order derivatives

$$\frac{\partial^2 \log p_{\theta_A}(\alpha_{A,k})}{\partial \theta_{A,i} \partial \theta_{A,j}} = - \frac{\delta_{ki} \delta_{kj}}{\theta_{A,k}^2} - \frac{\delta_{k|A|}}{\left(1 - \sum_{l=1}^{|A|-1} \theta_{A,l}\right)^2}. \quad (5.13)$$

The second order derivatives in Equation 5.13 pose a potential problem for the inversion of F . In the extreme cases where all samples have chosen the same production $\alpha_{A,k}$ the Fisher-information will be exactly equal to the negative of the respective Hessian. In that case, the Fisher-information will not be invertible, since it will be either a zero matrix with exactly one diagonal entry equal to one or a matrix of ones, $F = \mathbf{1}$. However, both of these extreme cases require that always the same production was sampled, implying a probability of 1.0 for that production, and, hence, a previous convergence of the nonterminal rule.

5.2 Experiments

We evaluated the presented approach on several task. First we illustrate how the method can learn grammar parameters in order to produce desired sentences and even completely eliminate undesired productions for a simple toy task. Afterwards, we trained a grammar to optimally play tic-tac-toe. Finally, a grammar for cleaning up the tic-tac-toe board using both robot arms was defined and the presented method was applied to learn which pairs of stones can be picked up concurrently while avoiding collisions between the arms.

5.2.1 Learning a Desired Length for $a^n b^n$

In the first task we apply the presented method to a grammar describing the language $a^n b^n$, where a and b are occasionally replaced by the terminal c. The goal of this task

is now to learn grammar parameters that produce sentences with predefined, desired average length, while simultaneously avoiding the undesired terminal symbol. The initial grammar

$$\begin{aligned}
\mathcal{G} &= \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle \\
\mathcal{A} &= \{a, b, c\}, \mathcal{V} = \{A\}, \mathcal{S} = \{A\} \\
\mathcal{R} &= \{ \\
&\quad \text{START} \rightarrow A \quad (1.00) \\
&\quad A \rightarrow aAb \quad (0.25) \\
&\quad \quad \rightarrow cAb \quad (0.25) \\
&\quad \quad \rightarrow aAc \quad (0.25) \\
&\quad \quad \rightarrow ab \quad (0.25) \\
&\quad \}
\end{aligned} \tag{GR 5.1}$$

produces sentences with an average length of 8 terminals, where each sentence can contain one or more cs. We define the exponential reward function

$$\text{reward}(\mathbf{T}) = \exp\left(-\frac{(|\mathbf{T}| - d)^2}{2}\right) - 0.1|\mathbf{T}|_c - 1.0$$

that prefers sentences with a desired length d that do not contain any cs. The terms $|\mathbf{T}|$ and $|\mathbf{T}|_c$ denote the number of terminals in general and the number of c-terminals respectively contained in the tree. The constant offset -1.0 ensures a maximal possible reward of 0.

Figure 5.2 shows several statistics for a desired length of $d = 10$. The evaluation ran for 100 iterations and at every iteration 500 samples or paths from the current grammar were drawn. Each sample consists of a single sentence, i.e., the horizon for each path is $H = 1$. Each plot shows the mean and two times the standard deviation over 50 trials. The variance of the learning curve in Figure 5.2a was expected, since the curve shows the return for each sample averaged over 500 samples but the goal of the task was to minimize the distance between the average length of the produced sentences $\text{avg}|\mathbf{T}|$ and d as shown in Figure 5.2b. Figure 5.2c shows the average number of c terminals per iteration. As expected the number decreases quickly until it reaches 0. Assuming each produced sentence is a sequence of actions, the reduction of c terminals exemplifies how the presented method can eliminate undesired actions. Generally the presented method adapts the grammar parameters such that a given objective is optimized. Figure 5.2d illustrates this behavior by showing the progression of each parameter value over 50 trials. As expected the parameters for the two productions that produce the undesired terminal c are trending towards 0 immediately. The now available probability mass is first assigned to the last parameter, which is due to the parameterization of $p(\alpha_{A,i})$. However, given that this distribution of the probability mass does achieve good results, the two remaining parameters are adjusted until the optimal parameterization is learned. The small parameter updates provided by the natural gradient method ensure a smooth trajectory for all parameters.

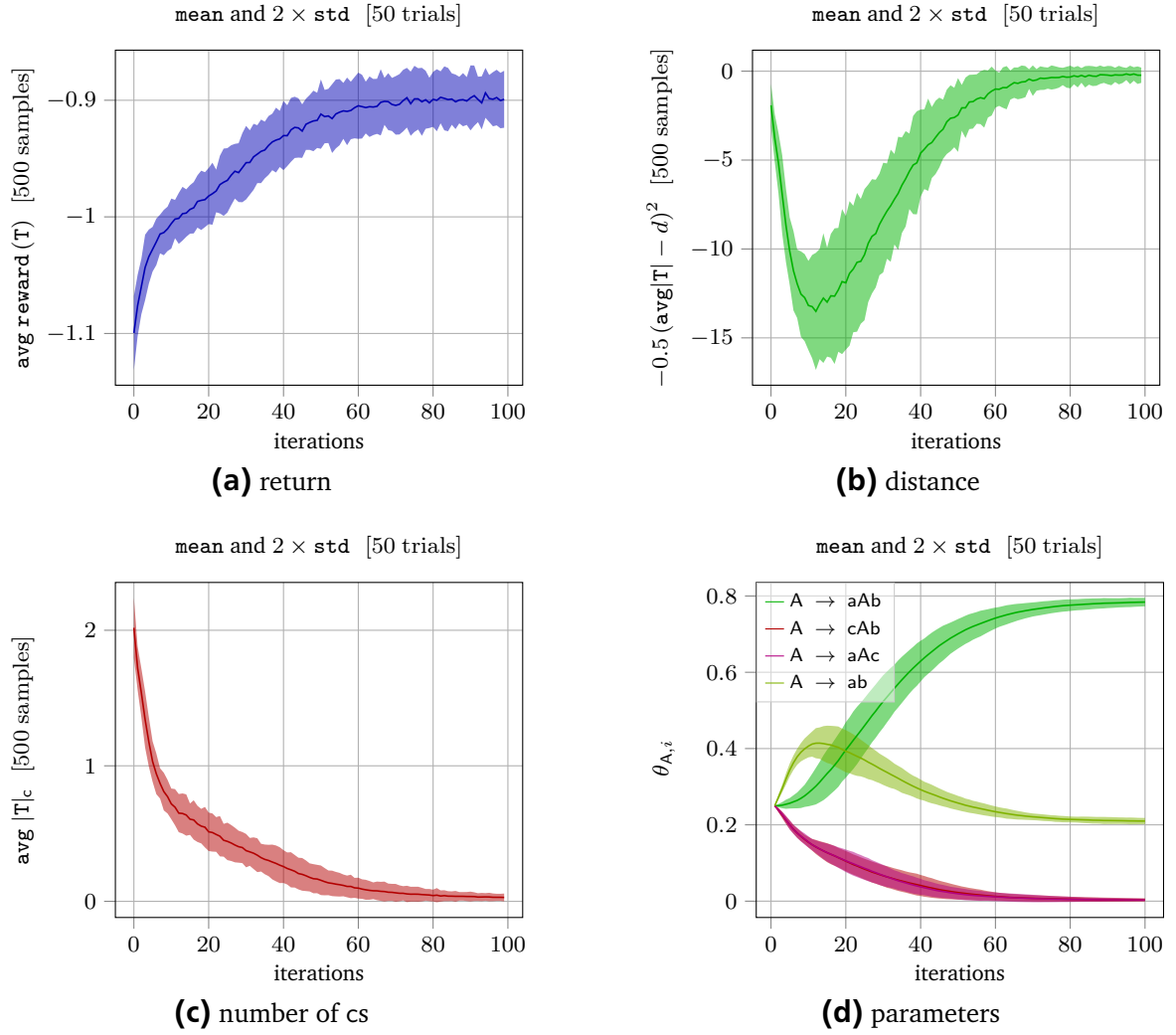


Figure 5.2.: The evaluations of the described $a^n b^n$ task for a desired average sentence length of $d = 10$ while minimizing the occurrences of c . **(a)** The average return over 500 samples per iteration. The mean and the standard deviation are computed over 50 trials. **(b)** The distance between the average sentence length per iteration and the desired length. **(c)** The average number of c s in the produced sentences. **(d)** The learned parameters for every iteration. The natural gradient method ensures smooth and steady trajectories. The trajectories for two rules producing the letter c overlap strongly.

5.2.2 Learning to Play Tic-Tac-Toe

In this scenario we aim to learn a grammar that plays tic-tac-toe. We focus on a learning a tic-tac-toe strategy for the first player, Player X. However, the same scheme can be applied to learn a strategy for the second player, Player O. We define 11 terminal symbols. One symbol for every field, one terminal representing the transposition of the state along the main diagonal and one terminal representing a rotation of the field by 90 degrees.

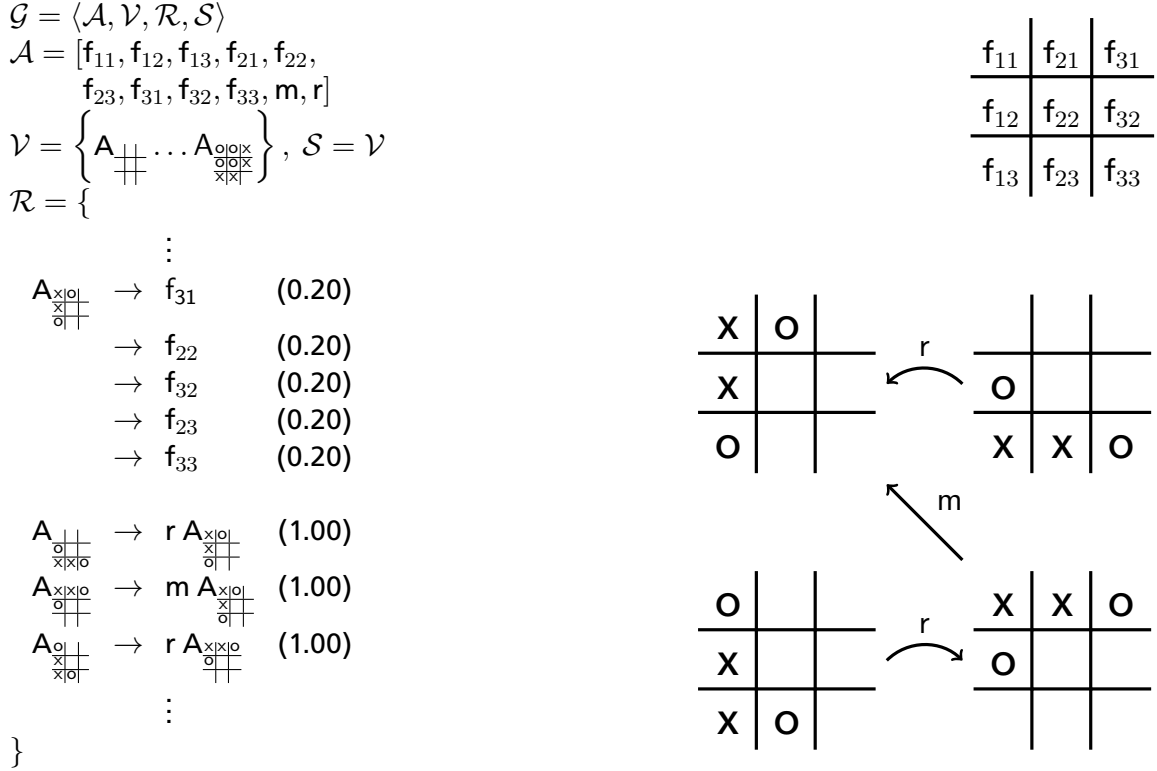


Figure 5.3.: Left: A part of the initial grammar for playing tic-tac-toe. Each state that requires a move from Player X is defined as a nonterminal. If a state is equal to the rotation or transposition of an already existing state, the only production represents the transformation between those two states. **Right:** An illustration of the terminal actions including the field names and the rotational and transpositional symmetries between different states of the tic-tac-toe playing field.

Each nonterminal encodes a possible state of the playing field that expects a move from Player X, i.e., a playing field with same number of xs and os. The productions for each nonterminal represent either empty fields that can be marked with a x or a transformation, rotation and transposition, into another state that is already present in the grammar. The left side of Figure 5.3 sketches such a grammar. The right side illustrates the meaning of the m and r terminals. For instance the rule for the second nonterminal in Figure 5.3 states that the field chosen by the first nonterminal has to be projected onto a 90 degree rotated field. Such rules allow the reduction of the parameter space significantly, since learning an optimal move for one state automatically defines the moves for all rotations and transpositions.

In contrast to the previous task a path τ consists of multiple sentences, where a sentence is defined by the leaves of a parse tree. In every path the grammar starts by choosing a field to place its marker on. Then the opponent makes its turn and the grammar chooses a production for the nonterminal corresponding to the current state of the playing field. Once no free field is left or one of the two players won, the path is finished. We define the reward for the entire path as

$$\text{reward}(\tau) = (6 - |\tau|)w,$$

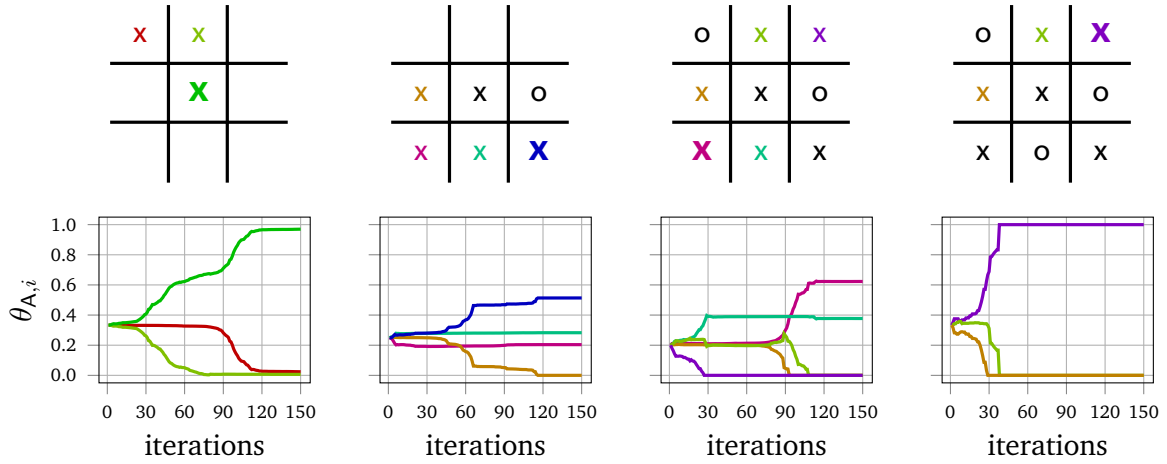


Figure 5.4.: The learned parameters of four different states of a match of tic-tac-toe after 150 iterations. The respective playing field is shown above each plot. Black marks illustrate already played moves, whereas the colored xs correspond to the learned parameters. The emphasized mark is the one with the highest probability after convergence.

where $|\tau|$ denotes the length of the path and w represents the result of the corresponding match. The result w is positive if the grammar won, negative if it lost and zero in case of a draw. Given that each τ has a maximal length of 5 the reward becomes higher if the grammar wins in few moves and lower if the grammar loses in few moves. Hence, the learned grammar should prefer short paths that yield in a win or if winning is not possible it should prolong the game as much as possible. The opponent plays a random strategy that always makes a winning move if possible otherwise it will always block a winning move of the grammar. If neither of those two options is available it will place its mark on a random free field. By playing against a random strategy the grammar will explore more of the possible state space than by playing against an optimal backwards induction strategy. Figure 5.4 shows a possible match between the final grammar and the random strategy. In addition, the figure shows the evolution of the parameters for the individual states over 150 iterations. For clarity, only a single trial is depicted in the parameter evolution. Given that the evolution of the parameters highly depends on the actions taken by an opponent with a random strategy, the actual parameter trajectories might look very different for each trial. Therefore, an illustration with mean and covariance is not informative, especially since the game of tic-tac-toe supports multiple optimal strategies. However, all



Figure 5.5.: The learned strategy was evaluated on several matches against a human opponent. The robot movement was given as a sequence of movement primitives produced by a movement grammar.

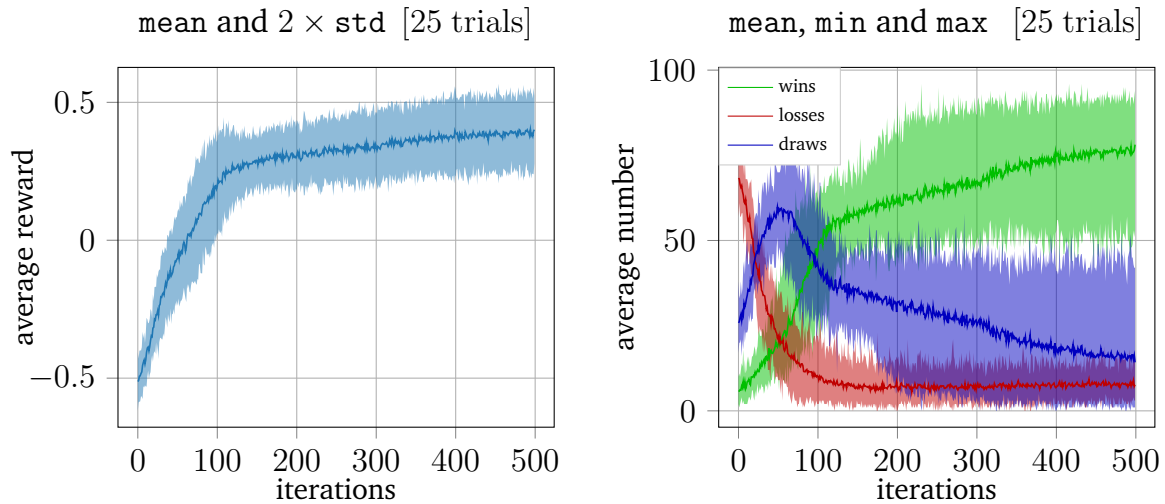


Figure 5.6.: Left: The average reward of 100 samples of playing tic-tac-toe against the described random strategy. The mean and two times the standard deviation are based on 25 trials **Right:** the average number of wins, draws and losses over 100 samples. The mean, the minimal and the maximal numbers were determined over 25 trials.

trials resulted in valid optimal solutions for the task. Suboptimal actions are quickly reduced and finally eliminated by the proposed approach. Figure 5.5 shows this particular match as it was played against a human opponent. The robot movement was encoded as a sequence of movement primitives, whereas the sequence was produced by formal grammar as introduced in (Lioutikov et al., 2018).

The left side of Figure 5.6 shows the average reward over 100 samples, while the right side depicts the average number of losses, wins and draws. In all cases the mean, the minimum and the maximum of the and two times the standard deviation are computed over 25 trials. The noisy trajectories are explained by the aforementioned random strategy of the opponent. Each sample in each trial therefore represents a very different match with different outcome. However, in total a clear tendency towards actions that minimize the losses and maximize the wins is evident. The reason why the number of losses does not reach zero yet is that despite the random strategy of the opponent the state space was not fully covered yet, resulting in potential losses for regions that have not yet been sufficiently explored.

5.2.3 Learning to Clean the Tic-Tac-Toe Board

The goal of this task is to clean up the tic-tac-toe playing field using both arms of the robot in as few steps as possible while avoiding collisions between the arms. The right arm picks up the x-markers and positions them back on a side panel, such that they are ready to be used by the robot again. The left arm picks up the o-markers placed by the human and throws them into a basket. We define a grammar that contains a nonterminal for every possible scenario. The productions for every nonterminal consist of three symbols, representing an action for the left arm, an action for the

right arm and the state of the playing field after both actions were executed. The actions for the left and the right arm simply indicate which stone should be picked up by the respective arm. Such a grammar is illustrated in Figure 5.7. Every possible state of the playing field is encoded as a nonterminal, the terminals to the left and to the right determine which actions will be executed simultaneously, by the left arm and right arm respectively. Given that the movements of the two arms are inherently different, this task in contrast to the previous one does not offer any symmetries that would reduce the state space of the playing field. Hence, the parameter space is significantly larger and the task is computationally very expensive.

To alleviate the computational complexity, we formulate the task as a curriculum learning problem. We begin by learning solutions for all states in which only one stone is left to be cleaned up. Once the learning converged for all such states, we start learning solutions for all playing fields where two stones are left. We then continue with 3 left stone and so on until we arrive at the states where the entire playing field is covered with stones. Applying this scheme, reduces the cleaning up task to a computationally feasible problem. The reward function for this task punishes three types of erroneous behaviors:

Collisions between the arms are predicted using a collision map projected onto the playing field. Given the action for the left and for the right arm the map indicates if a simultaneous execution would lead to a collision. A choice of action pairs for both arms that would result in a collision reduces the reward significantly.

Inefficiency regarding the action selection is punished. If the chosen action for one of the arms is to stay idle, f_{no} , even though there exists an action that would not lead to a collision the reward is reduced.

Sloppiness with respect to the cleaning task is also punished. If there are playing stones left on the field after the entire sentence has been executed, the reward is reduced further.

$$\begin{aligned}
 \mathcal{G} &= \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle \\
 \mathcal{A} &= [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, \dots \\
 &\quad f_{23}, f_{31}, f_{32}, f_{33}, f_{no}] \\
 \mathcal{V} &= \left\{ A_{\begin{smallmatrix} \text{+} & \text{+} \\ \text{x} & \text{o} \end{smallmatrix}} \dots A_{\begin{smallmatrix} \text{o} & \text{o} & \text{o} \\ \text{o} & \text{o} & \text{x} \\ \text{x} & \text{x} & \text{x} \end{smallmatrix}} \right\}, \mathcal{S} = \mathcal{V} \\
 \mathcal{R} &= \{ \\
 &\quad \vdots \\
 A_{\begin{smallmatrix} \text{x} & \text{o} \\ \text{x} & \text{+} \\ \text{o} & \text{+} \end{smallmatrix}} &\rightarrow f_{13} A_{\begin{smallmatrix} \text{o} \\ \text{+} \\ \text{+} \end{smallmatrix}} f_{11} \quad (0.11) \\
 &\rightarrow f_{13} A_{\begin{smallmatrix} \text{x} & \text{o} \\ \text{+} \\ \text{+} \end{smallmatrix}} f_{12} \quad (0.11) \\
 &\rightarrow f_{13} A_{\begin{smallmatrix} \text{x} & \text{o} \\ \text{x} & \text{+} \\ \text{o} & \text{+} \end{smallmatrix}} f_{no} \quad (0.11) \\
 &\rightarrow f_{21} A_{\begin{smallmatrix} \text{x} \\ \text{+} \\ \text{o} \end{smallmatrix}} f_{11} \quad (0.11) \\
 &\rightarrow f_{21} A_{\begin{smallmatrix} \text{x} \\ \text{+} \\ \text{o} \end{smallmatrix}} f_{12} \quad (0.11) \\
 &\rightarrow f_{21} A_{\begin{smallmatrix} \text{x} \\ \text{x} \\ \text{o} \end{smallmatrix}} f_{no} \quad (0.11) \\
 &\rightarrow f_{no} A_{\begin{smallmatrix} \text{o} \\ \text{x} \\ \text{+} \end{smallmatrix}} f_{11} \quad (0.11) \\
 &\rightarrow f_{no} A_{\begin{smallmatrix} \text{x} & \text{o} \\ \text{o} & \text{+} \\ \text{+} & \text{+} \end{smallmatrix}} f_{12} \quad (0.11) \\
 &\rightarrow f_{no} f_{no} \quad (0.11) \\
 A_{\begin{smallmatrix} \text{x} & \text{+} \\ \text{o} & \text{+} \end{smallmatrix}} &\rightarrow f_{13} f_{11} \quad (0.25) \\
 &\rightarrow f_{13} A_{\begin{smallmatrix} \text{x} \\ \text{+} \\ \text{+} \end{smallmatrix}} f_{no} \quad (0.25) \\
 &\rightarrow f_{no} A_{\begin{smallmatrix} \text{x} \\ \text{+} \\ \text{o} \end{smallmatrix}} f_{11} \quad (0.25) \\
 &\rightarrow f_{no} f_{no} \quad (0.25) \\
 &\quad \vdots \\
 &\}
 \end{aligned}$$

Figure 5.7.: A part of the initial tic-tac-toe cleaning grammar. Each possible playign field state is encoded as a single nonterminal. The actions to the left and to the right determine which field should be cleaned by the left and right arm respectively.

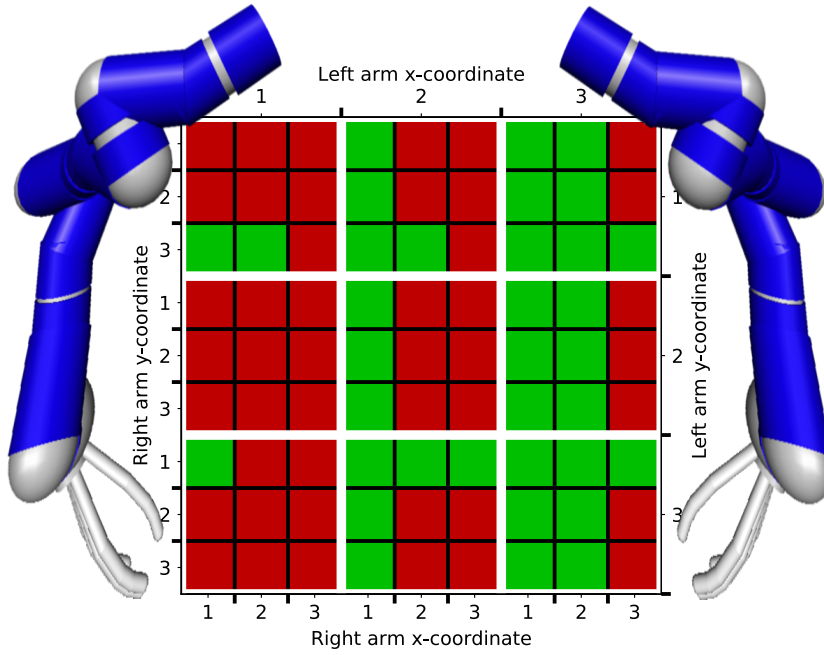


Figure 5.8.: The collision map for the tic-tac-toe cleaning up task. Red squares represent a collision, while green squares are collision free. For instance, if the left arm picks up a playing stone from field (1,3) the only collision free action for the right arm is to pick up a stone from field (1,1)

Figure 5.8 illustrates which, simultaneous actions of the left and right arm would lead to collisions. The outer, white grid separates the playing field into the possible actions for the left arm. For each chosen left arm action, the inner black grid corresponds to an action of the right arm. If the field is red a collision between the arms will occur. If the field is green, the respective actions can be executed simultaneously without a collision. Given the curriculum learning scheme and the collision

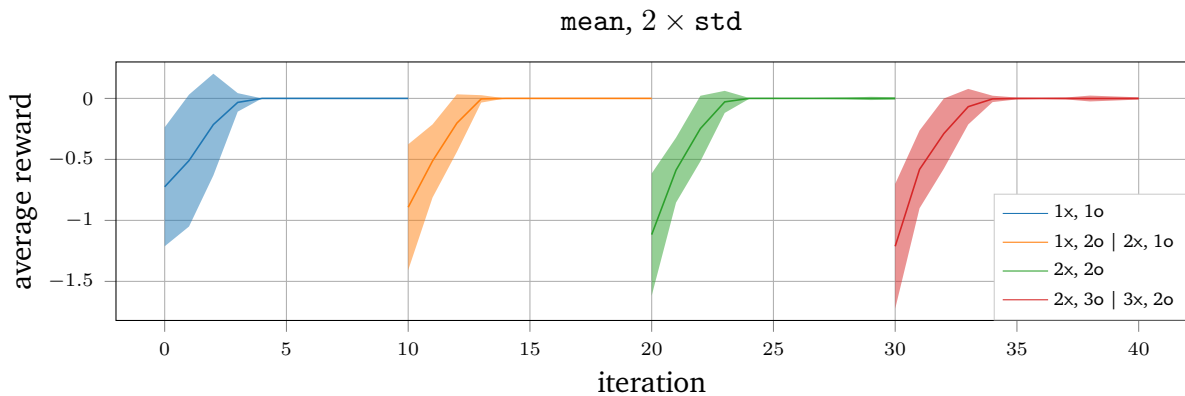


Figure 5.9.: The average reward over the different stages of the curriculum learning approach for the cleaning of the tic-tac-toe playing field. Once the previous stage convergence the learning of the next stage starts. The plots show the mean and two times the standard deviation of over all possible states of the corresponding stage.

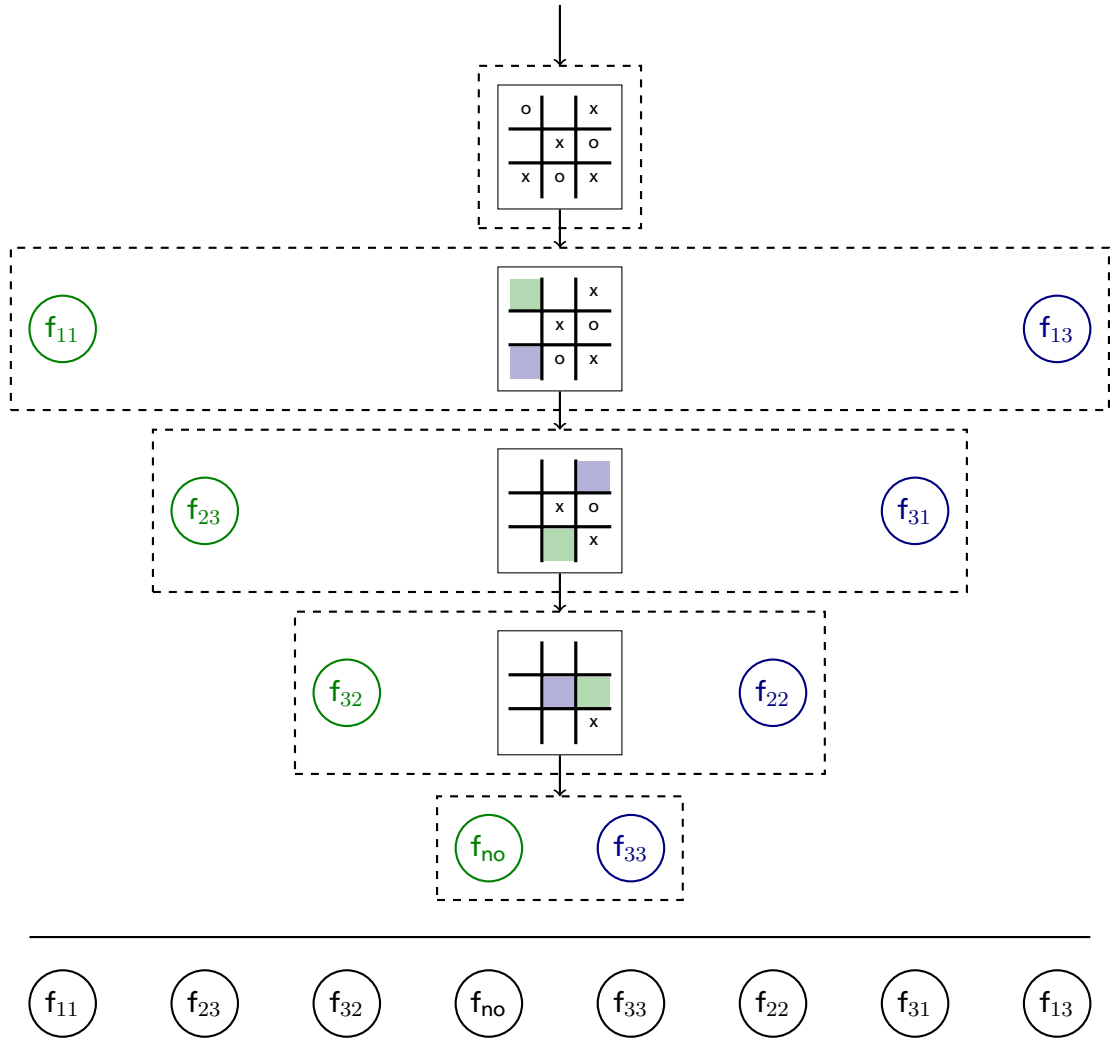


Figure 5.10.: The parse tree and the sequence to clean up a finished match of tic-tac-toe. The tree is compact and the produced sequence contains the maximum number of simultaneous action.

map the presented natural gradient method learned parameters that clean up the field reliably with a maximal number of simultaneous collision free left and right arm actions. Each stage of the curriculum scheme is started as soon as the previous stage converged to a solution, as shown in Figure 5.9. The graphs show the mean and two times the standard deviation of the first 4 stages. The individual stages converge after only few iterations. We ensure, that each stage is evaluated for at least 10 iterations. All possible playing fields that only contains a single marker type, either only xs or only os have trivial solutions and therefore no learning for these stage is required. We start the evaluation with the playing field that contains one x marker and one o marker, shown in blue. The learned grammar parameters ensures efficient primitive sequences, that clean every playing field with a maximal number of simultaneous actions. For instance, Figure 5.10 shows the parse tree and the sequence for cleaning up the final playing field of the match illustrated in Figure 5.4. The produced sequence is processed from both sides simultaneously, i.e., the left arm executes the left most action of the sequence, while the right arm executes the right most



Figure 5.11.: The learned grammar to clean up the tic-tac-toe playing-field was executed on a real robot platform. The field was cleaned in a minimal number of actions, while avoiding collisions.

action. The colors in the playing fields illustrated which field was cleaned by the left and right arm respectively. Figure 5.11 shows the execution of the learned grammar on a real robot platform.

5.3 Conclusion

In this chapter we introduced the concept of reinforcement learning to formal grammars. In contrast to common parameter estimation methods for formal grammars, the presented approach does not learn the parameters purely based on a batch of given observations. The grammar parameters are learned such that the produced parse trees, and, hence, the sequences maximize a given reward function. Given that the grammar parameters are the production probabilities for each rule, we ensured gradual parameter updates by applying the natural policy gradient method. We presented the log policy gradient, the corresponding Hessian and the Fisher-information matrix for formal grammars and pointed out potential caveats. The method was applied to learn a tic-tac-toe strategy formulated as a grammar and to a tic-tac-toe cleaning task, with the goal to clean the playing field as efficiently as possible. The presented method was able to learn grammar parameters that successfully solve both of these tasks.

Learning the production probabilities directly can cause issues with the convergence of the learned method and a careful choice of the learning rate. If the rate chosen to high the probability requirements might be violated after a parameter update. Therefore, we want to investigate ways to separate the learned parameters from the production probabilities. A potential approach is to learn the parameters of a function returning the concentration parameters of a Dirichlet distribution. Such an approach could also allow to formulate the grammar as a state dependent policy, instead of the currently assumed bandit setting.

6 Conclusion

In this thesis we presented an imitation learning pipeline consisting of three individual major contributions that each address different aspects of the pipeline. We finally provide a brief summary as well as an outlook into interesting future work and the most valuable lessons learned during the work on this Ph.D. thesis.

6.1 Summary

In Chapter 3 the Probabilistic Segmentation method was presented that segmented unlabeled demonstrations of an entire task into reoccurring patterns while simultaneously learning the underlying primitive library. The library was formulated as mixture of probabilistic movement primitives. ProbS is based on the Expectation-maximization approach, where the number of primitives is determined at every iteration through the G-means method. Therefore, the assignment of each segment and the mixture components are learned at the same time. The method was evaluated on several real robot tasks, whereas the demonstration were given either through kinesthetic teaching or motion capture. Hence, the method successfully segmented the demonstrations and learned a library of primitives in both, the joint and Cartesian space.

The segmented demonstration and the set of learned movement primitives were then used to induce a probabilistic context-free grammar. The presented induction method follows a Bayesian approach to grammar induction and assumes a grammar space in which all possible grammars are connected via operators. A Markov-chain Monte Carlo optimization searches for the grammar that optimizes a given posterior. We defined a novel prior for grammars, aimed at producing more comprehensible grammar structures. The new prior is based on three Poisson distributions centered around the number of rules, the number of production per rule and the number of symbols per production. The approach was evaluated on a tic-tac-toe pick-an-place task and a simple, collaborative box assembly task. In both task, the method successfully induced comprehensible grammars from an initial least-general conforming grammar.

In Chapter 5 we introduced the concept of reinforcement learning to formal grammars. In particular, we derived the natural policy gradient method for formal grammars. We successfully evaluated the method on a task to learn a grammar representing a tic-tac-toe playing strategy. In addition, we defined a grammar to clean up the tic-tac-toe field and applied the method to learn grammar parameters that produce efficient primitive sequences. In this task, the method learned parameters that avoid collisions between two potentially simultaneously active arm. Given that we use formal grammars as generative policies, the presented reinforcement learning can now be used in order to optimize the grammar for a news or modified task, without the need for additional, taught demonstrations.

In summary, this thesis provided several contributions to facilitate the interaction between humans and robots. The thesis does not claim to be a final solution to all human-robot collaboration challenges, but rather offers a framework aimed to relax the need for robotics expertise in human-robot interactions. The contributions presented in this thesis, resulted in several journal and conference publications, as listed below.

6.2 Future Work

The presented pipeline is a promising framework that easily allows for extensions of both the pipeline itself as well as the contained methods. An interesting challenge to be tackled in future work, for instance, is the approach to apply the induced grammar and the learned parameters as feedback to the segmentation process. This feedback would introduce another iterative layer to the entire pipeline aiming at further improved primitive libraries and movement grammars.

Another directly connected line of research is inverse reinforcement learning (IRL). Extending the pipeline with IRL methods would allow to deduct a reward function from the originally given demonstrations. The learned reward can then be used in the grammar reinforcement learning setup instead of a manually defined reward function. However, with a decreasing number of demonstrations it becomes less likely that the underlying reward can be learned accurately. This problem could be addressed by introducing active learning into the pipeline. After initially learning a reward function from few demonstrations, the agent executes several trajectories following the optimal policy given the learned reward. The new trajectories are then ranked by the collaborator and used to deduct a new reward that considers the ranking. Such an approach would allow to extrapolate better reward function from few demonstrations using inverse reinforcement and active learning.

A research direction extending this line of active reward learning is idea that ranked trajectories obtained by an optimal policy with respect to the current reward function might not always lead to the highest information gain. Hence, it might be advantageous to behave rather information optimal instead of following a policy that strictly optimizes the current reward. While the agent is still learning the reward function an information optimal behavior might result in ranked trajectories that allow the agent to learn a more detailed reward function and hence a better and potentially more robust policy. Future work will investigate how to learn information optimal policies and reward optimal policies simultaneously and co-dependently. Such an information optimal behavior can be further improved by considering the legibility of the executed trajectories. By executing motions that clearly highlight interesting areas of the current policy, it is easier for the collaborator to rank the demonstrations with respect to a desired and undesired behavior, resulting in a stronger correlation between the ranking and the underlying reward.

Another interesting line of research extending the pipeline is the concept of learning from observation. In contrast to learning from demonstration, learning from observation assumes that no actions or control signals are contained in the demonstrations. The observations consist solely of pure state space trajectories or even pixels of video frames. This would allow the agent to learn new tasks without ever having executed

the task before. In the future, we will research how to learn bidirectional mappings between primitive sequences and such observation spaces.

A significant shortcoming of the current pipeline is that both the segmentation and the grammar induction only consider the joint or Cartesian trajectories of the agent. Considering environment objects in these steps is, therefore, a promising direction for future research. For instance, changes in the dynamics of an object as well as relational changes with respect to other objects might indicate a change to a different subtask, and, hence, a transition point between movement primitives. Simultaneously, introducing task objects and their relations into the grammar induction adds a semantic layer to the process, allowing for the detection of different purposes of the same primitive, depending on the current state of the environment. Such a behavior can be modeled by a context-sensitive grammar, where the object features and relations would constitute the grammar contexts.

Including environment objects in the presented reinforcement learning approach is another promising idea. However, this extension would require a state-dependent modeling of the grammar, and, hence, the grammar parameters have to be conditional probabilities. A possible approach to achieve state-dependency would be to treat the grammar parameters as random variables drawn from non-symmetric Dirichlet distributions, where the concentrations parameters are defined by state dependent functions.

The environment can also be incorporated directly into the movement primitive representation in form of a context variable. In future work, we would like to investigate approaches that allow the agent to learn which task object features represent a context for each individual movement primitive. Such approaches can be extended by additionally learning task variables that are required for a successful execution, such as important, task-specific relations between certain objects. The environment can be incorporated into the primitive representation even further by including environment specific variables into the primitive state.

Other future extensions of movement primitives include the consideration of forces and temporal flexibility. Many tasks require a certain amount of applied forces at specific points during the task. However, the current probabilistic movement primitive representation does not consider forces applied during demonstrations. In future work, we will tackle this shortcoming by learning a force profile during the given demonstrations that allows to distinguish between desired forces and undesired perturbations during the execution of the primitive. Furthermore, we will investigate a state dependent phase to allow for variance along the temporal axis of a probabilistic movement primitive. Both of these approaches would be advantageous in collaborative human-robot interaction tasks. Hence, an extension to interaction primitives can be considered for the future as well.

An important learning aspect that is currently not addressed in the presented imitation learning pipeline is transfer learning. In the future, we would like to investigate methods to transfer behaviors learned from one task to new, previously unseen tasks. In particular, we are interested in improving the transfer learning capabilities of the pipeline, by including aspects of causality theory into the grammar induction process.

6.3 Lessons Learned

Developing the imitation learning pipeline did not only result in the presented methods, but also in a significant amount of experience. An important insight with respect to the segmentation process is the frame of reference of the trajectories. Some tasks are easier solved in Cartesian space while other tasks might be better represented in joint space. These state spaces differ significantly and might require an adaptation of the segmentation method in terms of initial heuristics and clustering procedure. However, finding or learning a latent manifold that combines the advantages of both without introducing other significant challenges is a Ph.D. thesis in itself. A general lesson that can be extracted here is to assess possible challenges, goals and solutions early on and then identify which steps are required to achieve the goal and which simplifications and assumptions can be made now and relaxed or removed later.

Another valuable lesson emerged during the development of the grammar induction approach. Grammar induction is considered a hard and unsolved problem. However, the context in which we operate, namely the learning of a grammar structure given sequences of movement primitives, changes the problem setting significantly from the common grammar induction problem. This allows the application of additional assumptions and constraints and eliminates possible caveats. At the same time other challenges arise, as for instance the connectivity requirements. Defining and considering the problem setting at hand highlights which aspects and findings of related work actually apply, revealing possible paths to tackle the problem.

Maybe the most important lesson learned during the course of this Ph.D. is that being a researcher requires the ability to find excitement in the research one is doing. Research does not consist of this one exciting problem that will change the world, but rather of a lot of smaller challenges that combined will result in something bigger than their sum. Tackling these challenges can result in a deviation from the personal research interests and be at times frustrating and unrewarding. However, being a researcher means to overcome such frustration and to find excitement in the given challenge. Excitement can be sparked through something as trivial as a new perspective or a new application.

Developing several approaches with the goal of a combined imitation learning pipeline provided a research experience with intermediate challenges that have been broken down into sub-goals themselves. Teaching me the valuable lesson that finding interesting research in challenges and problems that are part of a bigger picture instead of treating the underlying challenges as a black box leads to a research experience with less frustration and more excitement and gratification.

A Publication List

A.1 Journal Articles

Lioutikov, R. and J. Peters. “Reinforcement Learning for Formal Grammars”. In: *Advanced Robotics (AR) (submitted)* (2019)

Lioutikov, R., G. Maeda, F. Veiga, K. Kersting, and J. Peters. “Learning Attribute Grammars for Movement Primitive Sequencing”. In: *International Journal of Robotics Research (IJRR)* (2019)

Lioutikov, R., G. Neumann, G. Maeda, and J. Peters. “Learning Movement Primitive Libraries through Probabilistic Segmentation”. In: *International Journal of Robotics Research (IJRR)* 8 (2017), pp. 879–894

G. Maeda, G. Neumann, M. Ewerton, **Lioutikov, R.**, O. Kroemer, and J. Peters. “Probabilistic Movement Primitives for Coordination of Multiple Human-Robot Collaborative Tasks”. In: *Autonomous Robots (AURO)* 3 (2017), pp. 593–612

G. Maeda, M. Ewerton, G. Neumann, **Lioutikov, R.**, and J. Peters. “Phase Estimation for Fast Action Recognition and Trajectory Generation in Human-Robot Collaboration”. In: *International Journal of Robotics Research (IJRR)* 13-14 (2017), pp. 1579–1594

T. Osa, E. A. M. Ghalamzan, R. Stolkin, **Lioutikov, R.**, J. Peters, and G. Neumann. “Guiding Trajectory Optimization by Demonstrated Distributions”. In: *IEEE Robotics and Automation Letters (RA-L)* 2 (2017), pp. 819–826

A. Paraschos, **Lioutikov, R.**, J. Peters, and G. Neumann. “Probabilistic Prioritization of Movement Primitives”. In: *Proceedings of the International Conference on Intelligent Robot Systems, and IEEE Robotics and Automation Letters (RA-L)* (2017)

Lioutikov, R., A. Paraschos, J. Peters, and G. Neumann. “Generalizing Movements with Information Theoretic Stochastic Optimal Control”. In: *Journal of Aerospace Information Systems* 9 (2014)

A.2 Articles in Conference Proceedings

Lioutikov, R., G. Maeda, F. Veiga, K. Kersting, and J. Peters. “Inducing Probabilistic Context-Free Grammars for the Sequencing of Robot Movement Primitives”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 2018

D. Wilbers, **Lioutikov, R.**, and J. Peters. “Context-Driven Movement Primitive Adaptation”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2017

G. Maeda, A. Maloo, M. Ewerton, **Lioutikov, R.**, and J. Peters. “Anticipative Interaction Primitives for Human-Robot Collaboration”. In: *AAAI Fall Symposium Series. Shared Autonomy in Research and Practice, Arlington, VA, USA*. 2016

D. Koert, G. Maeda, **Lioutikov, R.**, G. Neumann, and J. Peters. “Demonstration Based Trajectory Optimization for Generalizable Robot Motions”. In: *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*. 2016

M. Ewerton, G. Neumann, **Lioutikov, R.**, H. Ben Amor, J. Peters, and G. Maeda. “Learning Multiple Collaborative Tasks with a Mixture of Interaction Primitives”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 2015, pp. 1535–1542

Lioutikov, R., G. Neumann, G. Maeda, and J. Peters. “Probabilistic Segmentation Applied to an Assembly Task”. In: *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*. 2015

G. Maeda, G. Neumann, M. Ewerton, **Lioutikov, R.**, and J. Peters. “A Probabilistic Framework for Semi-Autonomous Robots Based on Interaction Primitives with Phase Estimation”. In: *Proceedings of the International Symposium of Robotics Research (ISRR)*. 2015

A. Abdolmaleki, **Lioutikov, R.**, J. Peters, N. Lau, L. Reis, and G. Neumann. “Model-Based Relative Entropy Stochastic Search”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2015

Lioutikov, R., O. Kroemer, G. Maeda, and J. Peters. “Learning Manipulation by Sequencing Motor Primitives with a Two-Armed Robot”. In: *Intelligent Autonomous Systems 13 - Proceedings of the 13th International Conference IAS-13, Padova, Italy, July 15-18, 2014*. 2014, pp. 1601–1611

G. Maeda, M. Ewerton, **Lioutikov, R.**, H. Amor, J. Peters, and G. Neumann. “Learning Interaction for Collaborative Tasks with Probabilistic Movement Primitives”. In: *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*. 2014, pp. 527–534

Lioutikov, R., A. Paraschos, J. Peters, and G. Neumann. “Sample-Based Information-Theoretic Stochastic Optimal Control”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 2014

A.3 Workshop Papers

Lioutikov, R., F. Faller, M. Sigg, J. Perters, and G Maeda. “A Graph-Search Based Approach for Movement Primitive Sequencing”. In: *ICRA 2018, Abstract-Only Session*. 2018

Lioutikov, R., G. Maeda, F. Veiga, K. Kersting, and J Peters. “Learning Intuitive Grammars for Movement Primitive Sequences”. In: *ICRA 2018, Abstract-Only Session*. 2018

Lioutikov, R. and J Peters. “Movement Primitive Sequencing via Attribute Grammars”. In: *ICRA 2018, Third Machine Learning in Planning and Control of Robot Motion Workshop*. 2018

G. Maeda, A. Maloo, M. Ewerton, **Lioutikov, R.**, and J Peters. “Proactive Human-Robot Collaboration with Interaction Primitives”. In: *International Workshop on Human-Friendly Robotics (HFR)*. 2016

E. Rueckert, **Lioutikov, R.**, R. Calandra, M. Schmidt, P. Beckerle, and J. Peters. “Low-cost Sensor Glove with Force Feedback for Learning from Demonstrations using Probabilistic Trajectory Representations”. In: *ICRA 2015 Workshop on Tactile and force sensing for autonomous compliant intelligent robots*. 2015

M. Lopes, J. Peters, J. Piater, M. Toussaint, A. Baisero, B. Busch, O. Erkent, O. Kroemer, **Lioutikov, R.**, G. Maeda, Y. Mollard, T. Munzer, and D. Shukla. “Semi-Autonomous 3rd-Hand Robot”. In: *Workshop on Cognitive Robotics in Future Manufacturing Scenarios, European Robotics Forum, Vienna, Austria*. 2015

Lioutikov, R., O. Kroemer, J. Peters, and G Maeda. “Towards a Third Hand”. In: *IAS 13, 1st International Workshop on Intelligent Robot Assistants*. 2014

M. Ewerton, G. Neumann, **Lioutikov, R.**, H. Amor, J. Peters, and G Maeda. “Modeling Spatio-Temporal Variability in Human-Robot Interaction with Probabilistic Movement Primitives”. In: *ICRA 2014, Workshop on Machine Learning for Social Robotics*. 2014



B Curriculum Vitæ

Research Interest

Machine Learning:	Imitation Learning, Reinforcement Learning, Optimal Decision Making, Policy Search, Explainable AI, Causality Theory, Skill Acquisition, Movement Segmentation, Structure Learning, Grammar Induction, Skill Composition and Sequencing, Life-Long Learning, Active Learning
Robotics:	Anthropomorphic Robots, Human-Robot Interaction, Semi-Autonomy, Motor Skills, Movement Primitive Representation, Grasping, Manipulation, Adaptive Control, Human-In-The-Loop, Multi-Agent Systems, Robot Assisted Rehabilitation, Intelligent Prosthetics

Current Position as of 2019

01.2019 – **Assistant Professor of Practice**
Texas Institute for Discovery Education in Science
College of Natural Sciences
University of Texas at Austin, Austin, Texas, USA

Education

10.2013 – **Ph.D. student** at the Intelligent Autonomous Systems Group
10.2018 Technische Universität Darmstadt, Darmstadt, Germany

08.2015 – **Machine Learning Summer School**
09.2015 Kyoto, Japan

10.2011 – **Master of Science in Computer Science**
09.2013 Technische Universität Darmstadt, Darmstadt, Germany
Major in Robotics and Machine Learning. Minor in Bionics.
Thesis: "Learning time-dependent feedback policies with model-based policy search"
Advisors: Dr. tech. Gerhard Neumann, Prof. Dr. Jan Peters

07.2010 – **Exchange** program with the National University of the Province
09.2011 Buenos Aires, Tandil, Argentina

10.2006 – **Bachelor of Science in Computer Science**
07.2010 Technische Universität Darmstadt, Darmstadt, Germany

Honors and Awards

2019 **Georges Giralt Ph.D. Award Finalist**
European Robotics Forum, Bucharest, Romania

Research Experience

01.2019 – **Assitant Professor of Practice**
University of Texas at Austin

10.2013 – **Graduate Research Assistant**
10.2018 Intelligent Autonomous Systems Group
Technische Universität Darmstadt
Developing new methods to introduce robotics into small and medium
sized enterprises
3rd Hand Project - Seventh Framework Programme
(FP7-ICT-2013-10)

09.2012 – **Undergraduate Research Assistant**
09.2013 Intelligent Autonomous Systems Group
Technische Universität Darmstadt
Evaluation of robot learning methods on a throwing scenario

Teaching Experience

01.2019 – **Assitant Professor of Practice**, Robot Learning
University of Texas at Austin

10.2015 – **Teaching Assistant**, Robot Learning
04.2016 Technische Universität Darmstadt

04.2015 – **Teaching Assistant**, Intelligent Multi-Agent Systems
10.2015 Technische Universität Darmstadt

10.2014 – **Teaching Assistant**, Technical Foundations of Computer Science
04.2015 Technische Universität Darmstadt

04.2013 – **Lecture Assistant**, Software Engineering Design and Construction
09.2013 Technische Universität Darmstadt

Student Supervision

- 07.2019 – **FRI Fellowship:** "Self-Supervised Semantic Grounding of Movement
08.2019 Primitive Sequences"
Bhave, S.; Kodali, P.; O'Neil, C.; Trowbridge, I.
University of Texas at Austin
- 07.2019 – **FRI Fellowship:** "Self-Supervised Segmentation of Movement Primitive
08.2019 Sequences"
Bhave, R.; Hao, K.; Thomson, I.
University of Texas at Austin
- 04.2017 – **Student Research Project:** "Learning Grammars for Sequencing
10.2017 Movement Primitives"
Berninger, K.; Szelag, S.
Technische Universität Darmstadt
- 10.2016 – **Student Research Project:** "Probabilistic Trajectory Segmentation by
04.2017 Means of Hierarchical Dirichlet Process Switching Linear Dynamical
Systems"
Sieb, M.; Schultheis, M.; Szelag, S.
Technische Universität Darmstadt
- 01.2016 – **Master's Thesis:** "Context-driven Movement Primitive Adaptation"
07.2016 Wilbers, D.
Technische Universität Darmstadt
- 10.2015 – **Master's Thesis:** "Combining Human Demonstrations and Motion
04.2016 Planning for Movement Primitive Optimization"
Koert, D.
Technische Universität Darmstadt
- 04.2014 – **Student Research Project:** "Inverse Kinematics for Optimal Human-
04.2015 robot Collaboration"
Koert, D.
Technische Universität Darmstadt
- 04.2014 – **Student Research Project:** "Sequencing of Movement Primitives for
04.2015 Task- and Motion Planning", Sigg M.; Faller, F.
Technische Universität Darmstadt
- 10.2013 – **Student Research Project:** "Competitive Robot Pong"
04.2014 Koert, D.; Vandommele, T.
Technische Universität Darmstadt

Reviewing

2019	Advances in Neural Information Processing Systems (NeurIPS) Conference on Robot Learning (CoRL) AAAI Conference on Artificial Intelligence (AAAI)
2018	The IEEE/RSJ International Conference on Intelligent Robots and Systems The IEEE Robotics and Automation Letters
2017	The IEEE International Conference on Robotics and Automation The IEEE Robotics and Automation Letters The IEEE/RSJ International Conference on Intelligent Robots and Systems
2016	The IEEE International Conference on Robotics and Automation The International Journal of Robotics Research The IEEE/RSJ International Conference on Intelligent Robots and Systems The 25th International Joint Conference on Artificial Intelligence
2015	Automatica Autonomously Learning Robots Workshop at Advances in Neural Information Processing Systems (NIPS) The IEEE/RSJ International Conference on Intelligent Robots and Systems
2014	The IEEE/RSJ International Conference on Intelligent Robots and Systems

List of Figures

1.1. The imitation learning pipeline. The pipeline consists of the three methods presented in this thesis. The Probabilistic Segmentation approach introduced in Chapter 3, the grammar induction presented in 4 and the natural policy gradient method for formal grammars derived in Chapter 5. Chapter 2 gives a short introduction to probabilistic movement primitives and formal grammars.	4
2.1. Left: A regular grammar describing the language a^+b^+ . The corresponding parse tree for the sequence aabb is shown in Figure 2.2b. Right: A hidden Markov model equivalent to the regular grammar. Squares describe hidden states. Circles describe the emissions. Figure 2.2a shows an instantiated HMM for the sequence aabb.	7
2.2. Parse trees for the sequence aabb of (a) and (b) the hidden Markov model and the regular grammar shown in Figure 2.1. (c) shows the corresponding parse tree for the context-free grammar shown in Grammar 2.1.	8
3.1. The robot platform used for a chair assembly experiment. We used a seven DoF KUKA lightweight arm equipped with a five finger DLR HIT Hand II as end effector. The executed movement primitives were learned by segmenting human demonstrations.	11
3.2. An illustration of a possible segmentation. (a) shows a one dimensional, continuous observation. (b) shows four initially suggested transitions, illustrated as black bars, and the five resulting segments, if all transitions were true positives. (c) shows a possible segmentation. The fourth transition was identified as a false positive transition, illustrated as a gray bar. Two primitives m_1 and m_2 were learned from the resulting four segments.	13
3.3. The figure illustrates several segmentations of a one dimensional trajectory with four potential transition points, including the start and end of the trajectory. The indices of the segments denote at which transition point the segment begins and at which it ends. Assuming the segment $s_{1,2}$ is of interest, the blue segments occur alongside $s_{1,2}$ in at least one segmentation. The red segments contain $s_{1,2}$ and can therefore not occur in the same segmentation. The gray areas illustrate all possible preceding and succeeding segmentations of $s_{1,2}$, denoted as $\mathcal{D}_{s_{1,2}}^{\rightarrow}$ and $\mathcal{D}_{s_{1,2}}^{\leftarrow}$ respectively.	21

- 3.4. The factor graph corresponds to an observed trajectory shown in Figure 3.3. The nodes correspond to the different segments. The segment $s_{1,2}$ is of interest and the blue and red nodes correspond to the blue and red segments. All nodes directly connected to $s_{i,j}$ are considered its neighbors. Neighbors from above are predecessors and neighbors to the right are successors, e.g., $\mathcal{A}_{s_{1,2}}^{\text{pred}} = \{s_{0,1}\}$ and $\mathcal{A}_{s_{1,2}}^{\text{succ}} = \{s_{2,3}, s_{2,4}, s_{2,5}\}$. Additionally, the sets $\mathcal{A}^{\text{start}} = \{s_{0,1}, s_{0,2}, s_{0,3}, s_{0,4}, s_{0,5}\}$ and $\mathcal{A}^{\text{end}} = \{s_{0,5}, s_{1,5}, s_{2,5}, s_{3,5}, s_{4,5}\}$ contain all segments starting at the beginning of the trajectory or ending at the end of the trajectory respectively. 22
- 3.5. The factor graph illustrates the computation of the segment weighting $\alpha_{s_{i,j}}$, formulated as message passing. The $\delta_{s_{i,j}}^{\rightarrow}$ and $\delta_{s_{i,j}}^{\leftarrow}$ messages are the sums of the incoming $\gamma_{s_{0..i-1,i}}^{\rightarrow}$ and $\gamma_{s_{j,j+1..N}}^{\leftarrow}$ messages. The messages $\gamma_{s_{0..i-1,i}}^{\rightarrow}$ and $\gamma_{s_{j,j+1..N}}^{\leftarrow}$ are passed by the preceding and succeeding nodes respectively, $\mathcal{A}_{s_{i,j}}^{\text{pred}}$ and $\mathcal{A}_{s_{i,j}}^{\text{succ}}$. The weighting $\alpha_{s_{i,j}}$ is the product of the incoming forward and backward messages time $f(s_{i,j})/Z$, as described in Equation 3.15. 23
- 3.6. The experimental setup of the real robot writing task. The executed sequence is not restricted to three letter words, except by the whiteboard. The observed trajectories were demonstrated by kinesthetic teaching. Subsequently a movement primitive library was learned by segmenting the trajectories using ProbS. Finally sequences of the learned movement primitive library were executed on the real robot platform. 26
- 3.7. ProbS is able to reduce the number of active transitions significantly, leading to a compact representation of the observations. 27
- 3.8. The six movement primitives learned by ProbS. From left to right. 1: The letter “y” when proceeding an “a”. 2: The corpus of an “u”. 3: The tail of the letters “a” and “u” when not followed by an “a”. 4: The corpus of an “a”. 5: the letter “y” when not proceeding an “a”. 6: The tail of the letters “a” and “u” when followed by an “a”. 27
- 3.9. Comparison of ProbS, BP-AR-HMM, EM-GMM on the letter segmentation task. All methods resulted in a similar number of movement primitives (8, 6, 7), but the resulting segmentations differed significantly. Same color within one row indicates the assignment to the same movement primitive. 28
- 3.10.(left) The quality of the found segments is quantified by the number of bits each observations requires to be encoded. Less is better. (right) The learned movement primitives are evaluated using a LOOCV. ProbS shows the advantage of including the movement primitive learning in the segmentation process. 29

3.11. The first demonstrations explained by the three different methods. Different colors within each plot illustrated different movement primitives. ProbS explains the demonstration with four movement primitives, EM-GMM identified eight movement primitives in the demonstration and BP-AR-HMM separated the demonstration into a total of 9 segments. Some of the segments have a very short duration and are not visible in the plot.	30
3.12. The first three dimensions of the four movement primitives learned by ProbS to exemplify the skills, omitting the remaining dimensions. The dark line shows the mean and the shaded area corresponds to two times the standard deviation. The movement primitives are semantically meaningful, i.e., approaching a leg, showing the leg to a camera, approaching the seat and going back to the home position.	31
3.13. The first three dimensions, corresponding to the Cartesian position, of the six observations. Each observation demonstrates the insertion of a chair leg into a hole in the seat. The transitions determined by the initial heuristic are illustrated as dots.	32
3.14. Demonstration and execution of the chair assembly task. The human was tracked during the chair assembly. The observed demonstrations were subsequently segmented using ProbS. The movement primitives of the learned library were subsequently sequenced and executed on a robot platform to assemble the chair.	33
3.15. The first three joint states of the 16 demonstrations. The forehand and backhand swings occur randomly and are unequal in duration. Additionally, the duration between swings is not fixed and can be considered a waiting period.	33
3.16. Kinesthetic teaching of the barrett wam robot platform for the table tennis task. While one person throws table tennis balls towards the robot, another person moves the robot manually in order to return the balls. During the teaching all seven joints are recorded for the subsequent segmentation.	34
3.17. Mean and 2x std for each joint. The gray lines are the segments used to learn the primitives. For illustration purposes the is was normalized. This normalization is not used in the ProbS algorithm.	34
3.18. The four learned primitives for the robot table tennis task. The trajectories are shown in Cartesian space and were computed by applying the forward kinematics to the mean of each corresponding primitive. The brightness represents the velocity of the trajectory and the arrows indicate the direction of the movement. The grey lines show the original demonstration.	35
4.1. The robot executes a turn in the tic-tac-toe game, represented as a sequence of movement primitives. The sequence was generated by a probabilistic context-free grammar learned from previously labeled observations.	38

4.2.	The grammar space \mathcal{G} contains all valid grammars $\mathcal{G}^0 \dots \mathcal{G}^*$. The space is traversed by applying operators $op \in \mathcal{O} = \{\text{merge}, \text{split}, \text{chunk}, \text{insert}\}$ on the current grammar. For every operator op generating \mathcal{G}' from \mathcal{G} , there exists an \overline{op} that generates \mathcal{G} from \mathcal{G}' , e.g., $\overline{\text{merge}} = \text{split}$, $\overline{\text{chunk}} = \text{insert}$	42
4.3.	(a) shows the $ \text{mean} + 2 \text{ std}$ of the velocity distribution defined in Equation 4.1 for each ProMP, while (b) shows the maximal velocities as computed in Equation 4.2. The colors indicate which categories were assigned to each ProMP and background colors highlight to which category each joint belongs.	43
4.4.	The transition overlap for each arm primitives of the tic-tac-toe task. A value above the threshold $\epsilon_{\text{overlap}} = 0.69$ signifies that the primitive at that row is connectible to the primitive of that column. The ellipsoids were $n = 2$ standard deviations wide and the threshold was determined from the observations with $\alpha = 0.95$	45
4.5.	The five arm primitives used in the sequences, representing turns in the tic-tac-toe game. While both <code>pick_near</code> and <code>pick_far</code> approach a stone from the home position they differ in the stone positions they can reach. Similarly the primitives <code>place_left</code> and <code>place_right</code> position the stone in different areas of the playing field.	58
4.6.	The posteriors and the likelihood for the tic-tac-toe turn grammar. The vertical, dashed line indicates the index of the highest posterior (171), given the presented Poisson prior.	59
4.7.	A parse tree of a sequence produced by the learned grammar for tic-tac-toe turns. Nonterminals are presented as squares and terminals as circles. A dashed rectangle represents the production chosen by the parent terminal with the probability next to the connecting arrow. The solid line separates the final sequence from the producing parse tree. The grammar was enhanced with the presented attributes and evaluation scheme, where stone and field are two keys assigned to the keywords attributes of the <code>pick_far</code> and <code>place_right</code> terminals respectively.	60
4.8.	The arm primitives of the box assembly task. The robot applies different primitives for grasping a board, <code>take_board</code> , or picking a screw, <code>take_screw</code> . Similarly the handover for boards and screws is encode in different primitives.	61
4.9.	The posteriors and the likelihood for the box assembly task. The vertical, dashed line indicates the index of the highest posterior (160), given the presented Poisson prior.	62
5.1.	In this chapter we formulate the natural policy gradient method for formal grammars. We learn the grammar parameters to learn a strategy for playing tic-tac-toe and a strategy to clean the board bi-manually while avoiding collisions between the arms.	65

5.2.	The evaluations of the described $a^n b^n$ task for a desired average sentence length of $d = 10$ while minimizing the occurrences of c. (a) The average return over 500 samples per iteration. The mean and the standard deviation are computed over 50 trials. (b) The distance between the average sentence length per iteration and the desired length. (c) The average number of cs in the produced sentences. (d) The learned parameters for every iteration. The natural gradient method ensures smooth and steady trajectories. The trajectories for two rules producing the letter c overlap strongly.	71
5.3.	Left: A part of the initial grammar for playing tic-tac-toe. Each state that requires a move from Player X is defined as a nonterminal. If a state is equal to the rotation or transposition of an already exiting state, the only production represents the transformation between those two states. Right: An illustration of the terminal actions including the field names and the rotational and and transpositional symmetries between different states of the tic-tac-toe playing field.	72
5.4.	The learned parameters of four different states of a match of tic-tac-toe after 150 iterations. The respective playing field is shown above each plot. Black marks illustrate already played moves, whereas the colored xs correspond to the learned parameters. The emphasized mark is the one with the highest probability after convergence.	73
5.5.	The learned strategy was evaluated on several matches against a human opponent. The robot movement was given as a sequence of movement primitives produced by a movement grammar.	73
5.6.	Left: The average reward of 100 samples of playing tic-tac-toe against the described random strategy. The mean and two times the standard deviation are based on 25 trials Right: the average number of wins, draws and losses over 100 samples. The mean, the minimal and the maximal numbers were determined over 25 trials.	74
5.7.	A part of the initial tic-tac-toe cleaning grammar. Each possible playign field state is encoded as a single nonterminal. The actions to the left and to the right determine which field should be cleaned by the left and right arm respectively.	75
5.8.	The collision map for the tic-tac-toe cleaning up task. Red squares represent a collision, while green squares are collision free. For instance, if the left arm picks up a playing stone from field (1,3) the only collision free action for the right arm is to pick up a stone from field (1,1) .	76
5.9.	The average reward over the different stages of the curriculum learning approach for the cleaning of the tic-tac-toe playing field. Once the previous stage convergence the learning of the next stage starts. The plots show the mean and two times the standard deviation of over all possible states of the corresponding stage.	76
5.10.	The parse tree and the sequence to clean up a finished match of tic-tac-toe. The tree is compact and the produced sequence contains the maximum number of simultaneous action.	77

5.11. The learned grammar to clean up the tic-tac-toe playing-field was executed on a real robot platform. The field was cleaned in a minimal number of actions, while avoiding collisions.	78
---	----

Bibliography

- Andrieu, C., N. de Freitas, A. Doucet, and M. I. Jordan. “An Introduction to MCMC for Machine Learning”. In: *Machine Learning* 50.1-2 (2003), pp. 5–43.
- Baker, J. K. “Trainable grammars for speech recognition”. In: *The Journal of the Acoustical Society of America* 65.S1 (1979), S132–S132.
- Barbič, J., A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard. “Segmenting motion capture data into distinct behaviors”. In: *Proceedings of the Graphics Interface 2004 Conference, May 17-19, 2004, London, Ontario, Canada*. Canada: Canadian Human-Computer Communications Society, 2004, pp. 185–194.
- Bishop, C. M. *Pattern recognition and machine learning*. Vol. 1. Springer New York, 2006.
- Brand, M. and V. Kettner. “Discovery and segmentation of activities in video”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 844–851.
- Bringmann, K. and T. Friedrich. “Approximating the volume of unions and intersections of high-dimensional geometric objects”. In: *Computational Geometry* 43.6 (2010), pp. 601–610. ISSN: 0925-7721.
- Chiang, D., A. K. Joshi, and D. B. Searls. “Grammatical Representations of Macromolecular Structure”. In: *Journal of Computational Biology* 13.5 (2006), pp. 1077–1100.
- Chiappa, S. and J. Peters. “Movement extraction by detecting dynamics switches and repetitions”. In: *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*. USA: Curran Associates, Inc., 2010, pp. 388–396.
- Chomsky, N. “Three models for the description of language”. In: *IRE Transactions on Information Theory* 2.3 (1956), pp. 113–124.
- Dantam, N., P. Kolhe, and M. Stilman. “The Motion Grammar for Physical Human-Robot Games”. In: *International Conference on Robotics and Automation*. IEEE, 2011.
- Dantam, N. and M. Stilman. “The Motion Grammar Calculus for Context-Free Hybrid Systems”. In: *American Control Conference*. 2012, pp. 5294–5301.

-
- Dantam, N. and M. Stilman. “The Motion Grammar: Analysis of a Linguistic Method for Robot Control”. In: *IEEE Trans. Robotics* 29.3 (2013), pp. 704–718.
- d’Avella, A. and M. C. Tresch. “Modularity in the motor system: decomposition of muscle patterns as combinations of time-varying synergies”. In: *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. USA: MIT Press, 2001, pp. 141–148.
- Earley, J. “An Efficient Context-Free Parsing Algorithm (Reprint)”. In: *Commun. ACM* 26.1 (1983), pp. 57–61.
- Endres, D., Y. Meirovitch, T. Flash, and M. A. Giese. “Segmenting sign language into motor primitives with Bayesian binning”. In: *Frontiers in Computational Neuroscience* 7 (2013), p. 68.
- Fod, A., M. J. Matarić, and O. C. Jenkins. “Automated derivation of primitives for movement classification”. In: *Autonomous Robots* 12.1 (2002), pp. 39–54.
- Fox, E. “Bayesian Nonparametric Learning of Complex Dynamical Phenomena”. Ph.D. Thesis. Cambridge, MA: MIT, 2009.
- Gold, E. M. “Language identification in the limit”. In: *Information and Control* 10.5 (1967), pp. 447–474.
- Hamerly, G. and C. Elkan. “Learning the k in k-means”. In: *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*. USA: MIT Press, 2003, pp. 281–288.
- Ijspeert, A. J., J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. “Dynamical movement primitives: learning attractor models for motor behaviors”. In: *Neural computation* 25.2 (2013), pp. 328–373.
- Ijspeert, A. J., J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. “Dynamical movement primitives: learning attractor models for motor behaviors”. In: *Neural computation* 25.2 (2013), pp. 328–373.
- Kakade, S. M. “A natural policy gradient”. In: *Advances in neural information processing systems*. 2002, pp. 1531–1538.
- Kitani, K. M., Y. Sato, and A. Sugimoto. “Deleted interpolation using a hierarchical Bayesian grammar network for recognizing human activity”. In: *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*. 2005, pp. 239–246.

-
- Kitani, K. M., Y. Sato, and A. Sugimoto. "Recovering the Basic Structure of Human Activities from Noisy Video-Based Symbol Strings". In: *IJPRAI* 22.8 (2008), pp. 1621–1646.
- Konidaris, G., S. Kuindersma, R. Grupen, and A. Barto. "Robot learning from demonstration by constructing skill trees". In: *The International Journal of Robotics Research* 31 (2012), pp. 360–375.
- Kormushev, P., S. Calinon, and D. G. Caldwell. "Robot motor skill coordination with EM-based Reinforcement Learning". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*. IEEE, 2010, pp. 3232–3237. ISBN: 978-1-4244-6674-0.
- Krishnan, S., A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg. "Transition State Clustering: Unsupervised Surgical Trajectory Segmentation For Robot Learning". In: *Proceedings of International Symposium on Robotics Research, Sep 12-15, 2015, Genova, Italy*. 2015.
- Kroemer, O., H. van Hoof, G. Neumann, and J. Peters. "Learning to Predict Phases of Manipulation Tasks as Hidden States". In: *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*. IEEE, 2014, pp. 4009–4014.
- Kulic, D., C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura. "Incremental learning of full body motion primitives and their sequencing through human motion observation". In: *I. J. Robotics Res.* 31.3 (2012), pp. 330–345.
- Kulic, D., W. Takano, and Y. Nakamura. "Online segmentation and clustering from continuous observation of whole body motions". In: *IEEE Transactions on Robotics* 25.5 (2009), pp. 1158–1166.
- Lee, K., T. Kim, and Y. Demiris. "Learning action symbols for hierarchical grammar induction". In: *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012*. IEEE Computer Society, 2012, pp. 3778–3782. ISBN: 978-1-4673-2216-4.
- Lee, K., Y. Su, T. Kim, and Y. Demiris. "A syntactic approach to robot imitation learning using probabilistic activity grammars". In: *Robotics and Autonomous Systems* 61.12 (2013), pp. 1323–1334.
- Lemme, A., F. Reinhart, and J. J. Steil. "Semi-supervised Bootstrapping of a Movement Primitive Library from Complex Trajectories". In: *Proceedings of the International Conference on Humanoid Robots 2014, Nov 18-20, 2014, Madrid, Spain*. IEEE, 2014, pp. 726–732.
- Lenneberg, E. H. "The biological foundations of language". In: *Hospital Practice* 2.12 (1967), pp. 59–67.

-
- Leonard, H. C. and E. L. Hill. “Review: The impact of motor development on typical and atypical social cognition and language: a systematic review”. In: *Child and Adolescent Mental Health* 19.3 (2014), pp. 163–170.
- Lioutikov, R., G. Maeda, F. Veiga, K. Kersting, and J. Peters. “Learning Attribute Grammars for Movement Primitive Sequencing”. In: *International Journal of Robotics Research (IJRR)* (submitted) (2018).
- Lioutikov, R., G. Neumann, G. Maeda, and J. Peters. “Learning Movement Primitive Libraries through Probabilistic Segmentation”. In: *International Journal of Robotics Research (IJRR)* 8 (2017), pp. 879–894.
- Lioutikov, R., G. Neumann, G. Maeda, and J. Peters. “Learning Movement Primitive Libraries through Probabilistic Segmentation”. In: *International Journal of Robotics Research (IJRR)* 8 (2017), pp. 879–894.
- Lioutikov, R., G. Neumann, G. Maeda, and J. Peters. “Probabilistic Segmentation Applied to an Assembly Task”. In: *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*. 2015. URL: http://www.ausy.tu-darmstadt.de/uploads/Site/EditPublication/lioutikov_humanoids_2015.pdf.
- Lioutikov, R., A. Paraschos, J. Peters, and G. Neumann. “Sample-Based Information-Theoretic Stochastic Optimal Control”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 2014.
- Lioutikov, R. and J. Peters. “Reinforcement Learning for Formal Grammars”. In: *IEEE Robotics and Automation Letters (RA-L)* (submitted) (2018).
- Manschitz, S., J. Kober, M. Gienger, and J. Peters. “Learning to sequence movement primitives from demonstrations”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*. IEEE, 2014, pp. 4414–4421.
- Marcus, G. F. “Negative evidence in language acquisition”. In: *Cognition* 46.1 (1993), pp. 53–85.
- Meier, F., E. Theodorou, F. Stulp, and S. Schaal. “Movement segmentation using a primitive library”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*. IEEE, 2011, pp. 3407–3412.
- Muelling, K., J. Kober, O. Kroemer, and J. Peters. “Learning to Select and Generalize Striking Movements in Robot Table Tennis”. In: 3 (2013), pp. 263–279.
- Mülling, K., J. Kober, and J. Peters. “Learning Table Tennis with a Mixture of Motor Primitives”. In: *10th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2010, Nashville, TN, USA, December 6-8, 2010*. IEEE, 2010, pp. 411–416.

-
- Nakanishi, J., J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. "Learning from demonstration and adaptation of biped locomotion". In: *Robotics and Autonomous Systems* 47.2 (2004), pp. 79–91.
- Nakazawa, A., S. Nakaoka, K. Ikeuchi, and K. Yokoi. "Imitating human dance motions through motion structure analysis". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, September 30 - October 4, 2002*. Vol. 3. IEEE, 2002, pp. 2539–2544.
- Niekum, S., S. Chitta, B. Marthi, S. Osentoski, and A. Barto. "Incremental semantically grounded learning from demonstration". In: *Robotics: Science and Systems IX, Technische Universität Berlin, Berlin, Germany, June 24 - June 28, 2013*. Vol. 9. 2013.
- Niekum, S., S. Osentoski, G. Konidaris, and A. G. Barto. "Learning and Generalization of Complex Tasks from Unstructured Demonstrations". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*. IEEE, 2012, pp. 5239–5246.
- Niekum, S., S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. "Learning grounded finite-state representations from unstructured demonstrations". In: *The International Journal of Robotics Research* 34.2 (2015), pp. 131–157.
- Ondracek, J. M. and R. H. Hahnloser. "Advances in understanding the auditory brain of songbirds". In: *Insights from Comparative Hearing Research*. Springer, 2013, pp. 347–388.
- Paraschos, A., C. Daniel, J. Peters, and G. Neumann. "Probabilistic Movement Primitives". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. USA: Curran Associates, Inc., 2013, pp. 2616–2624.
- Paraschos, A., C. Daniel, J. Peters, and G. Neumann. "Using Probabilistic Movement Primitives in Robotics". In: (2017).
- Riedmiller, M. and H. Braun. *RPROP - A Fast Adaptive Learning Algorithm*. Tech. rep. Proc. of ISCIS VII), Universitat, 1992.
- Rivas, E. and S. R. Eddy. "The language of RNA: a formal grammar that includes pseudoknots". In: *Bioinformatics* 16.4 (2000), pp. 334–340.
- Rohrer, B. and N. Hogan. "Avoiding spurious submovement decompositions II: a scatter-shot algorithm". In: *Biological Cybernetics* 94.5 (2006), pp. 409–414.
- Sarabia, M., K. Lee, and Y. Demiris. "Towards a synchronised Grammars framework for adaptive musical human-robot collaboration". In: *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 2015, pp. 715–721.

-
- Smith, N. A. and J. Eisner. “Contrastive estimation: Training log-linear models on unlabeled data”. In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 2005, pp. 354–362.
- Stolcke, A. “Bayesian Learning of Probabilistic Language Models”. PhD thesis. Berkeley, CA, USA, 1994.
- Stulp, F. and S. Schaal. “Hierarchical reinforcement learning with movement primitives”. In: *2011 11th IEEE-RAS International Conference on Humanoid Robots*. 2011, pp. 231–238.
- Takano, W. and Y. Nakamura. “Humanoid robot’s autonomous acquisition of proto-symbols through motion segmentation”. In: *2006 6th IEEE-RAS International Conference on Humanoid Robots, Genova, Italy, December 4-6, 2006*. IEEE, 2006, pp. 425–431.
- Talton, J. O., L. Yang, R. Kumar, M. Lim, N. D. Goodman, and R. Mech. “Learning design patterns with bayesian grammar induction”. In: *The 25th Annual ACM Symposium on User Interface Software and Technology, UIST ’12, Cambridge, MA, USA, October 7-10, 2012*. ACM, 2012, pp. 63–74. ISBN: 978-1-4503-1580-7.
- Thelen, E. “Development as a Dynamic System”. In: *Current Directions in Psychological Science* 1.6 (1992), pp. 189–193.
- Wächter, M. and T. Asfour. “Hierarchical Segmentation of Manipulation Actions based on Object Relations and Motion Characteristics”. In: *International Conference on Advanced Robotics, ICAR 2015, Istanbul, Turkey, July 27-31, 2015*. IEEE, 2015, pp. 549–556.
- Williams, B., M. Toussaint, and A. J. Storkey. “Modelling motion primitives and their timing in biologically executed movements”. In: *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. USA: Curran Associates, Inc., 2008, pp. 1609–1616.
- Wu, C. F. J. “On the convergence properties of the EM algorithm”. In: *The Annals of Statistics* 11 (1983), pp. 95–103.
- Yamane, K., Y. Yamaguchi, and Y. Nakamura. “Human motion database with a binary tree and node transition graphs”. In: *Autonomous Robots* 30.1 (2011), pp. 87–98.
- Zhu, S.-C. and D. Mumford. “A stochastic grammar of images”. In: *Foundations and Trends® in Computer Graphics and Vision* 2.4 (2007), pp. 259–362.